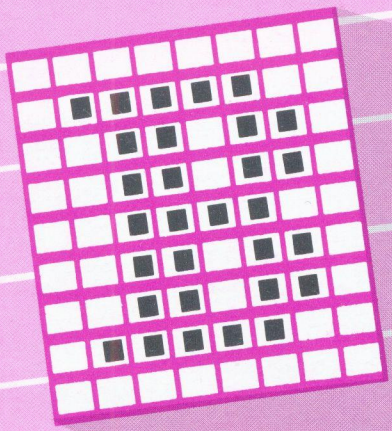
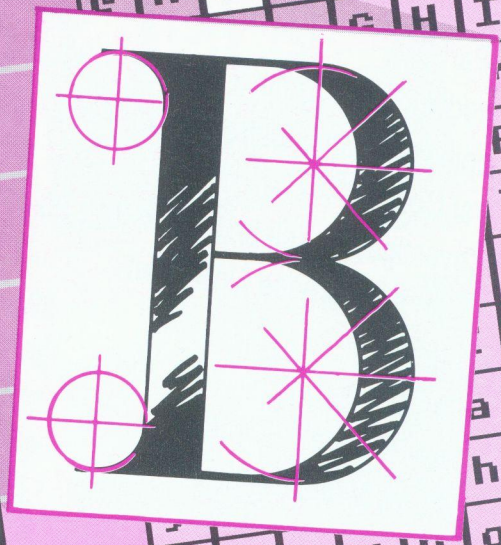
Vol.8 No.4 August/September 1989

BEEBUG

FOR THE BBC MICRO & MASTER SERIES





```
f0 Store char.
f1 Load char.
f2 Invert window
f3 Rotate window
f4 Mirror window
f5 Clear wndw/char
f6 Print chars
f7 Wipe whole font
f8 Exit
```

Font Designer

- LINE-INPUT FUNCTION
- COPING WITH COMPUTERS
- INTRODUCING POSTSCRIPT
- SELF-HELP UTILITY

BEEBUG Vol.8 No.4 August/September 1989

FEATURES

REVIEWS

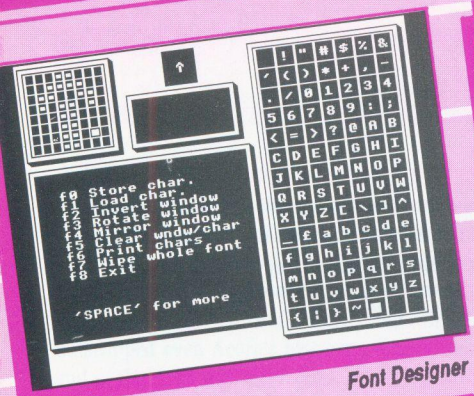
REGULAR ITEMS

HINTS & TIPS

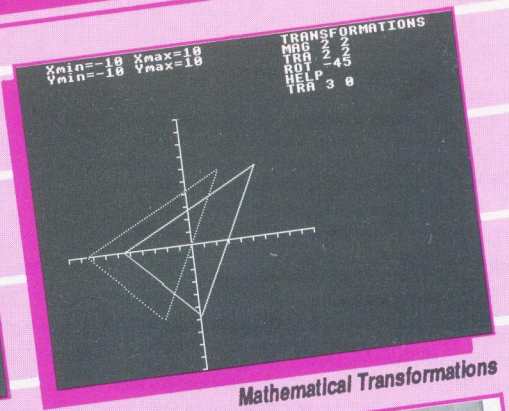
PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

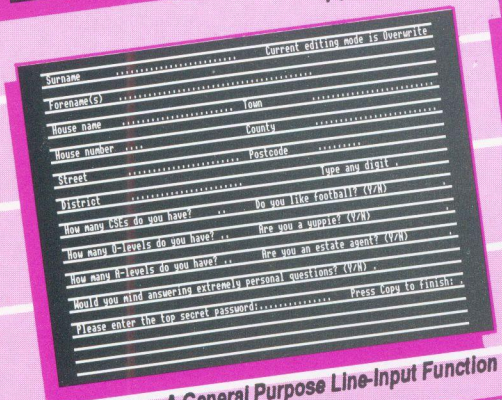
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



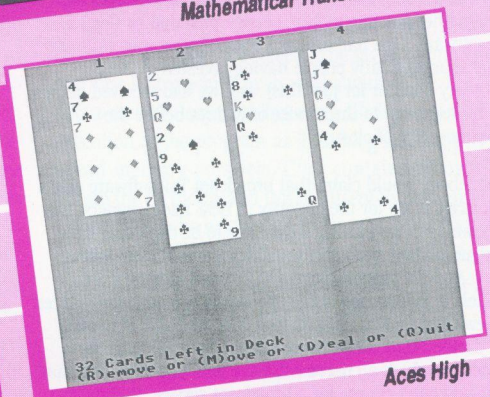
Font Designer



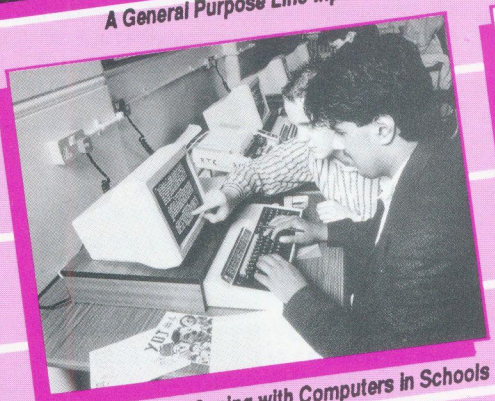
Mathematical Transformations



A General Purpose Line-Input Function



Aces High



Coping with Computers in Schools

Multi-Colour Printing

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

THE DEMISE OF CEEFAX TELESOFTWARE

It is not often that an event provokes strong criticism, but the sudden and blunt announcement at the end of July that the Telesoftware service provided by the BBC as part of Ceefax was to be axed at the end of August, just one month later, has certainly caused anguish. It was only in Vol.8 No.2 (June 1989) that John Woodthorpe reported on the new weather satellite picture service provided by Ceefax Telesoftware. This had aroused considerable interest from many sources (many people having invested in the necessary adaptor for just this), and we had received details of improvements to the service only days before the announcement of closure.

Now nobody would claim that broadcast Telesoftware was a major fact of life for the majority of BBC micro owners, but the manner of its parting surely has to be deplored, particular when such exciting developments were taking place. And the manufacturers of Teletext adaptors are unlikely to be amused either. No doubt someone well placed in News and Current Affairs (the current home of Ceefax) has decided that telesoftware doesn't 'pay' (despite the licence fee which all users will have paid anyway), and that since you can't charge for a broadcast service, it shouldn't be done at all.

This hardly seems like support for the BBC micro, from the sales of which the BBC continues to receive a substantial royalty, and we can only deplore the apparent haste and unfeeling manner in which the decision to close down the service has been taken.

If you wish to express your own views on this matter we would suggest you write to the Director of News and Current Affairs, BBC TV Centre, Wood Lane, London W12 7RJ, or phone 01-743 8000.

BEEBUG READER SURVEY

At the time of writing well over 250 replies have been received, and more are arriving daily. We are obviously delighted at the high level of response, and we will shortly begin the task of analysing the results in order to plan for the future. Many thanks to all those who took the time and effort to answer all our questions.

A3000 ON SPECIAL OFFER TO TEACHERS

Acorn has announced a joint scheme with selected Acorn dealers (including BEEBUG) to offer the A3000 at a reduced price to teachers. Full details are available from BEEBUG.

BEEBUG HAS MOVED

Please remember that we now have a new address,

117 Hatfield Road, St Albans, Herts AL1 4JS.

Our telephone numbers stay the same though. Our Open Day at the new building is Sunday 10th September 1989.

LOOKING AHEAD WITH BEEBUG

The following features are planned for the October issue of BEEBUG:

Applications and Utilities:
- Graph Plotting
- Printer Control Commands
- Font Designer Part 2
- Auto-Input for Assembler
- Disc File Identifier

Reviews:
- Master Control ROM
- DOS+ Problem Solver

Plus First Course, Workshop, News, Hints and more.

RISC USER

RISC User is the largest circulation magazine devoted exclusively to the Acorn Archimedes range of computers. It is available on subscription to all BEEBUG members at a substantially reduced rate (see facing inside back cover).

We expect the October issue to contain:
- Desktop Diary Part 2
- ARM Code Single Stepper Part 2
- Icon Selector Shell
- Colour Image Processing
- Assembler Workshop
- Mastering the Wimp Part 2

Reviews:
- Pipedream 3
- Armadeus
- Forms and Labels Designer
- Dabs Press Instigator

and more.

RISC User is the ideal magazine to keep you up to date with all that's happening in the Archimedes world, and is particularly useful if you are contemplating the purchase of an Archimedes or A3000 in the near future.

News News News News News News

A3000 STEALS SHOW

There is little doubt that the most popular attraction at the recent BBC Acorn User Show held at Alexandra Palace was the new Acorn A3000. The new machine is not only fully compatible with the latest Archimedes systems, but is also the new official 'BBC Micro'. Sales of the A3000 outstripped even Acorn's best expectations, and by the end of the three day show there was only a handful of machines left unsold. Part of the success must be due to the scheme devised by Acorn and Mercantile Credit which allowed people to buy the A3000 with 0% finance over one year. As an added incentive Acorn were offering a free monitor stand and carrying case, and BEEBUG backed this up with a package of bonus items worth nearly £100.

Furthermore, a number of companies were already demonstrating hardware add-ons and software for the A3000. Complete compatibility with a RISC OS-based Archimedes means that there should be no shortage of software for the A3000.

Meanwhile, even though everybody's eyes are on the A3000, Acorn have pledged to support the existing Master 128, and earlier Beebs, for at least two years. However, the turbulent life of the Master Compact is being ended, with Acorn stopping all production in January 1990.

MUSICAL DISCS

AMPLE DCT, the support service for users of the Hybrid Music 500 and Music 5000 synthesizers, has released a series of discs containing music files. For the Music 5000 there are six discs, with at least twelve tunes on each, while for the Music 500 the entire Oxygene and Equinox suites are available. There are also a set of three discs containing data files for use with the Island Music editor. Each disc features 25 tunes. All the discs are 5.25" 80 track format, and cost £5 each (inc. VAT). For more details contact Panda Discs, Four Seasons, Tinkers Lane, Brewood, Stafford ST19 9DE. Alternatively, details can be found on the Ample DCT bulletin board which can be contacted on (0384) 239944 (Viewdata), or (0384) 238073 (Scrolling).

CLICK CLICK

Slogger Computers have made a comeback into the Acorn market with *Click* - a pop-up utility supplied in a ROM cartridge for the Master 128. The idea is that at the click of a button (hence the name), the current task is suspended and *Click's* menu pops up. This allows access to several features including a full calendar and diary, a snapshot utility which allows the screen to be saved or printed, a file and disc editor which even works with ADFS hard discs, and a memory editor. After using *Click* you can continue execution of the program as if nothing had happened. To avoid memory conflicts, *Click* contains its own RAM within the cartridge, and this is battery backed-up to allow information to be held for up to three months. *Click*, which is only suitable for the Master 128, costs £59.95 (inc. VAT), and is available from Slogger Computers, 7 Apsley Road, Clifton, Bristol BS8 2SH, tel. (0272) 743683.

NEW UNIX WORKSTATION

Acorn are working on a successor to their R140 UNIX workstation. The new machine, which will be released in the near future will be based around the new ARM 3 processor, and will have 8Mbyte of RAM, twice that of the R140. A major criticism of the original machine was that UNIX could barely run on a 4Mbyte computer. The ARM 3 processor is fully compatible with the existing ARM 2, but includes an on-chip cache allowing very high processor speeds while still using standard RAM. There is no indication of the price of the new machine, but it will almost certainly be outside the range of the home or secondary education user.

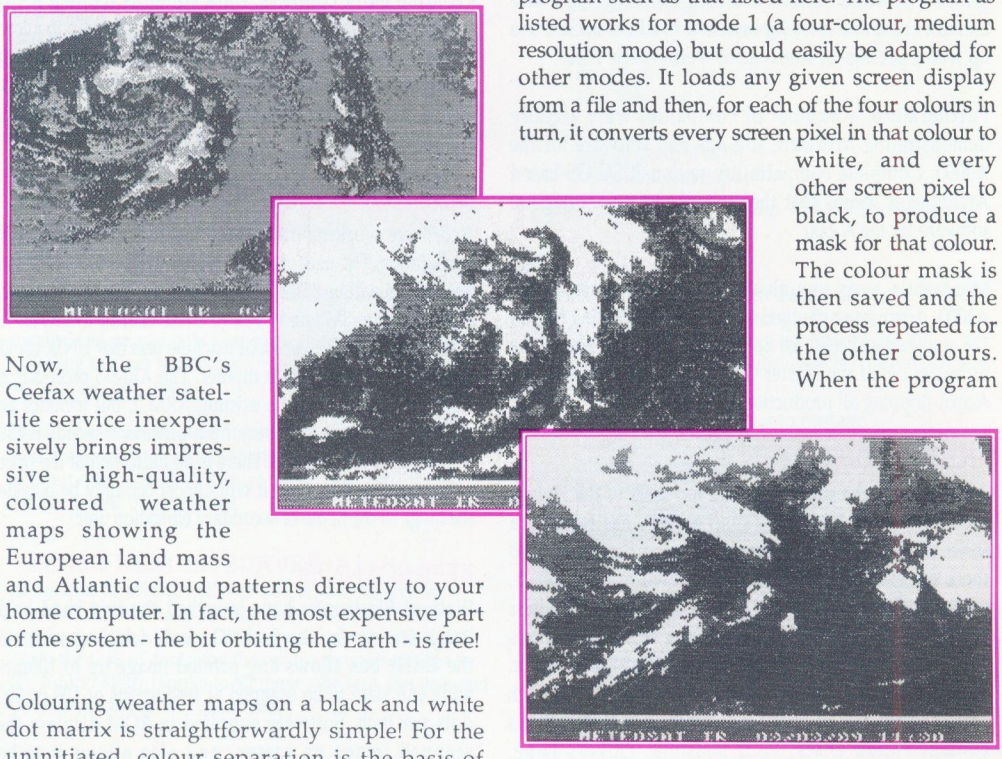
BEEB SCANNER

Watford Electronics have launched the first hand-held image scanner for the Beeb. The unit, which connects to the 1MHz bus allows any printed image up to 105mm wide (A6 size) to be scanned at resolutions of 100 or 200 dots per inch. Software supplied in ROM displays the scanned image on screen, and also allows it to be transferred directly into Watford's *Wapping Editor* package. The Beeb Hand-Scanner costs £155.25 (inc. VAT), and is available from Watford Electronics, Jessa House, 250 Lower High Street, Watford, tel. (0923) 37774.

Multi-Colour Printing

Dorian Goring explains how you can produce multi-coloured printout, even if you only have an ordinary dot-matrix printer.

The fascination of information technology (IT) is that, like life itself, it never ceases to offer something new (see John Woodthorpe's article on downloading weather satellite pictures in BEEBUG Vol.8 No.1). This is especially true if you're interested in exploring graphics and colour separation techniques. And this doesn't mean you have to have a colour printer! Weather maps form an excellent subject for demonstrating this technique.



Images taken by METEOSAT, printed with different colour ribbons

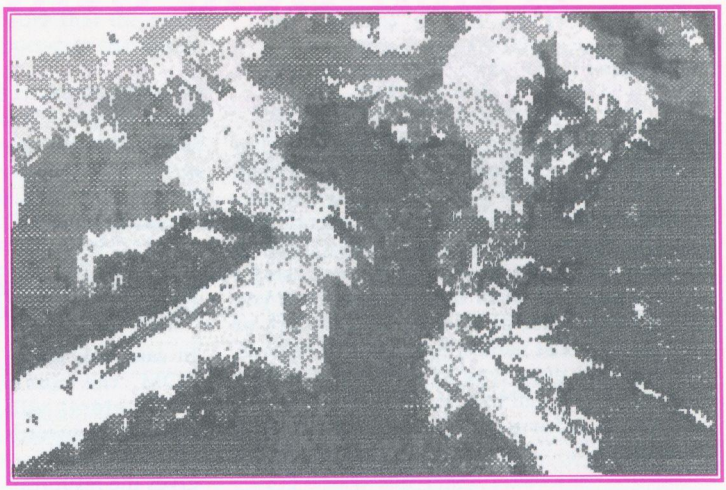
Now, the BBC's Ceefax weather satellite service inexpensively brings impressive high-quality, coloured weather maps showing the European land mass and Atlantic cloud patterns directly to your home computer. In fact, the most expensive part of the system - the bit orbiting the Earth - is free!

Colouring weather maps on a black and white dot matrix is straightforwardly simple! For the uninitiated, colour separation is the basis of colour printing. The three primary colours - yellow, magenta, cyan - together with black are printed separately, but in combination produce a wide range of hues and tints.

Coloured ribbons are available for most printers, or you can remove the ribbon and use coloured carbons face-to-face with white or tinted paper. What you need to do is to create separate masks - one for each colour - from the original multi-coloured image. This is freehand editing, and is best done within a graphics or desktop publishing (DTP) environment (such as AMX Art or StopPress).

Alternatively you can use a simple splitter program such as that listed here. The program as listed works for mode 1 (a four-colour, medium resolution mode) but could easily be adapted for other modes. It loads any given screen display from a file and then, for each of the four colours in turn, it converts every screen pixel in that colour to white, and every other screen pixel to black, to produce a mask for that colour. The colour mask is then saved and the process repeated for the other colours. When the program terminates, we have four new screen dumps saved, one for each of the four colours in mode 1.

For other modes, the main loop will need to to cycle from 0 to 1, 0 to 2, or 0 to 15 depending upon the number of colours per mode. The step size in line 200 (but not 190) can usefully be changed to 8 in modes 2 and 5 (low resolution) and to 2 in mode 0 (high resolution) to save time.



You will need to use a suitable screen printer dump routine (e.g. BEEBUG's Dumpmaster) to print each screen in turn. The colour separation process means that one sheet of paper now makes several passes through the printer, each time with a different colour mask and ribbon (or carbon). You'll find that you need registration marks for accurate printing, and you should use lighter colours first then darker ones to minimise "muddying" ribbons.

Colours force the eye to move round the map creating movement. This mirrors the movement of swirling masses of vast clouds as they move day by day around the planet.

Careful colour balancing allows the creation of intriguing and dazzling effects, which, literally, appear to vibrate before your eyes! Also, discordant colours in juxtaposition create visual conflict and tension, again, reflecting the powerful forces at work.

The rewarding point about weather pictures in particular is that they bridge the gulf between aesthetics and science. They are a product of the era of information technology created by the logical and highly organised power of the computer. However the techniques for multi-coloured printing described above can be applied to almost any multi-coloured screen display.

NOTE: We are sorry that we are unable to do full justice to the multi-coloured printouts which Dorian Goring submitted with his article, because of the limitations of two-colour printing as used in BEEBUG.

```
  10 REM Program Split
  20 REM Version B1.0
  30 REM Author  Dorian Goring
  40 REM BEEBUG  August/September 1989
  50 REM Program subject to copyright
  60 :
 100 MODE1
 110 *.
 120 INPUT'"Filename: " pic$
 130 VDU19,0,0,0,0,0
 140 VDU19,1,1,0,0,0
 150 VDU19,2,4,0,0,0
 160 VDU19,3,7,0,0,0
 170 FOR colour%=0 TO 2
 180 OSCLI("LOAD "+pic$)
 190 FOR y%=0 TO 1023 STEP 4
 200 FOR x%=0 TO 1279 STEP 4
 210 IF POINT(x%,y%)<>colour% THEN VDU1
8,0,7 ELSE VDU18,0,0
 220 PLOT69,x%,y%
 230 NEXT x%
 240 NEXT y%
 250 OSCLI("SAVE "+pic$+CHR$(48+colour%
)+" 3000+5000")
 260 NEXT colour%
 270 END
```

A Self-Help Utility

Bernard Hill's comparatively short program will enable you to install a comprehensive help system on your computer, customised to your own individual needs.

This article describes a simple help facility which you can implement on your own system. This is in the form of a ROM image which may be loaded into sideways RAM, or programmed into an EPROM for permanent installation. It allows for up to nearly 16K of help text, using a simple system of recall by keyword. Thus you could keep a list of telephone numbers or addresses against names; catalogues of records, or you could even type in sections of system help such as FX call lists or PLOT numbers - or any other information you find hard to memorize.

Once installed you could simply type:

```
*ASK FX5
```

or:

```
*ASK HARRY
```

to produce the information you have previously stored under the keywords 'FX5' or 'HARRY'.

HOW TO MAKE YOUR INFORMATION ROM

First you will need to create a help text file with Wordwise, Edit or View, etc. Avoid any embedded characters or commands, just as if you were building an EXEC file (although the program will handle the 'soft' spaces put in by View when justifying text). Keywords in the text should be in upper-case and surrounded by '£' signs as in this example:

```
£MARY£
953-4476
£JOHN£
44053  (day)
01-234-5678 (evenings)
£BARRY£
4321 Ask for Ext. 221
```

Here 'MARY', 'JOHN' and 'BARRY' are keywords; the text then follows each keyword and may run (as in 'JOHN') to as many lines as you like. Thus, in this example:

```
*ASK BARRY
```

would produce the following:

```
4321 Ask for Ext 221
```

and:

```
*ASK JOHN
```

would give:

```
44053  (day)
01-234-5678 (evenings)
```

Of course, the text could be much longer than these few words, and typically might fill one screen.

In case you forget what is contained in your ROM, the command *ASK (without any parameters) will give a complete list of available keywords, thus:

```
*ASK
```

gives the response:

```
MARY
JOHN
BARRY
```

in this trivial example.

CREATING YOUR ROM IMAGE

Having typed in and saved the program, and created your text file, run the program. The only prompt is for the filename of the text file. This is then coded into a ROM image and saved on disc as a file called ASKROM. This can be loaded into sideways RAM with *SRLOAD (or the command appropriate to your sideways RAM board), and used as already described.

LIMITATIONS AND CUSTOMISATION

Since '£' is the delimiter for keywords in the text, it clearly cannot be used in the text. If this is a problem then you can define a different delimiter, such as '\'. Simply replace the variable *sep* in line 120 of the program, and make sure that the only entries in the text file which contain this symbol are the keyword delimiters. The keywords should be in upper-case only and contain no spaces.

If *ASK is a command already used by a different ROM in your computer, or if you wish to have a second self-help ROM in your computer, then you can change the command by altering the variable *command$* in line 110. You could use command$="Q", perhaps, but don't use:

```
command$="HELP"
```

for obvious reasons. Make sure that the name assigned to *command$* is in upper-case characters.

Other customisable features in the program are the variable *image$* in line 110, which is the name of the ROM image under which the ROM is saved on disc, and the variable title$ in line 120 which is the name of the ROM as you want it to appear in response to a *ROMS command.

On this month's program disc you will find a 15K information file for you to use which contains a wealth of help on PLOT, COLOUR, SOUND, etc.

```
 10 REM Program Ask ROM
 20 REM Version B1.0
 30 REM Author  Bernard Hill
 40 REM BEEBUG  August/September 1989
 50 REM Program subject to copyright
 60 :
100 MODE7:HIMEM=&3C00
110 command$="ASK":image$="ASKROM"
120 title$="ASK ROM":sep=ASC"£"
130 FOR opt=4 TO 6 STEP 2
140 P%=&8000:O%=&3C00
150 [ OPT opt:BRK:BRK:BRK
160 JMP serve
170 EQUB &82:EQUB copy:EQUB 0
180 EQUS title$
190 .copy EQUB 0
200 EQUS "(C) Beebug, 1989":EQUB 0
210 .name EQUS command$
220 .serve:CMP #4:BNE ret
230 PHA:TYA:PHA:LDX#0
240 .loop LDA (&F2),Y:AND #&DF
250 CMP name,X:BNE notme
260 CPX #LENcommand$-1:BEQ eon:INX
270 INY:JMP loop
280 .eon INY:LDA (&F2),Y:CMP #13
290 BNE over:JMP bare
300 .over:CMP #32:BEQ keyword
310 .notme PLA:TAY:PLA:LDX &F4
320 .ret RTS
330 .keyword INY:LDA (&F2),Y:CMP #32
340 BEQ keyword:CMP #13:BNE over0
350 JMP bare:.over0
360 LDA #(text+1) MOD 256:STA &A8
370 LDA #(text+1) DIV 256:STA &A9
380 TYA:CLC:ADC &F2:STA &F2:BCC over1
390 INC &F3
400 .over1:LDA &F2:STA &AA
410 LDA &F3:STA &AB
420 .try:LDA &AA:STA &F2:LDA &AB
430 STA &F3:LDY #0
440 .comp:LDA (&A8),Y:BEQ eow
450 LDA (&F2),Y:AND #&DF:CMP (&A8),Y
460 BNE notword:INY:JMP comp
470 .eow LDA (&F2),Y:CMP #13
480 BEQ found:CMP #32:BEQ found
490 .notword:LDA (&A8),Y:BEQ nextkey
500 CMP #&FF:BEQ end
510 INY:BNE notword:INC &A9
520 JMP notword
530 .nextkey:INY:BNE over2:INC &A9
540 .over2 LDA (&A8),Y:BNE nextkey
550 CMP #&FF:BEQ end
560 .nextfound:INY
570 TYA:CLC:ADC &A8:STA &A8:BCC try
580 INC &A9:JMP try
590 .found:INY:BNE over3:INC &A9
600 .over3:LDA (&A8),Y:BEQ fin
610 JSR &FFE3:JMP found
620 .fin JSR &FFE7
630 .end PLA:TAY:PLA:LDA #0:LDX &F4
640 RTS
650 .bare LDA #text MOD 256:STA &A8
660 LDA #text DIV 256:STA &A9:LDY #1
670 .loop:LDA (&A8),Y:BEQ eok
680 CMP #&FF:BEQ end:JSR &FFE3:INY
690 BNE loop:INC &A9:JMP loop
700 .eok:JSR &FFE7
710 .loop2:INY:BNE over4:INC &A9
720 .over4:LDA (&A8),Y:CMP #&FF
730 BEQ end:CMP #0:BNE loop2:INY
740 BNE loop:INC &A9:JMP loop
750 .text
760 ]
770 NEXT
780 spare=&C000-text-1
790 PRINTspare" bytes for text"'
800 REPEAT
810 INPUT "Filename of text : "name$
820 f=OPENINname$
830 IF f=0 THEN PRINT" --- file not fo
und"
840 UNTIL f>0
850 ext=EXT#f:CLOSE#f
860 IF ext>spare THEN PRINT"File too l
arge, ROM not formed":END
870 OSCLI("LOAD "+name$+" "+STR$~(text
-&4400))
880 PRINT"Please wait - arranging text
:"
890 FOR I%=text-&4400 TO text-&4400+ex
t
900 IF ?I%=sep THEN ?I%=0
910 IF ?I%=26 THEN ?I%=32
920 NEXT
930 !(text-&4400+ext)=&FF00
940 OSCLI("SAVE "+image$+" 3C00 7C00 8
000 8000")
950 PRINT'"    ROM saved as "image$
960 PRINT'spare-ext" bytes spare capac
ity."
970 END
```

Font Designer

Ian Stewart presents the first part of a comprehensive and professional font designer for the BBC micro.

The characters that you see on the screen of your computer system, are represented as numbers internally. Most computer systems, including the BBC micro, use a standard coding system called ASCII (American Standard Code for Information Interchange). In turn, the pattern of dots (8 by 8) which represents the shape of each character on the screen, is represented by a further set of eight numbers.

The style of the character on the screen is called a font. However, the BBC micro allows the user to change the definition for each character, and thus new fonts may be created. The purpose of the programs listed here, and concluded next month, are to provide a simple but powerful application which overcomes most of the difficulties normally encountered.

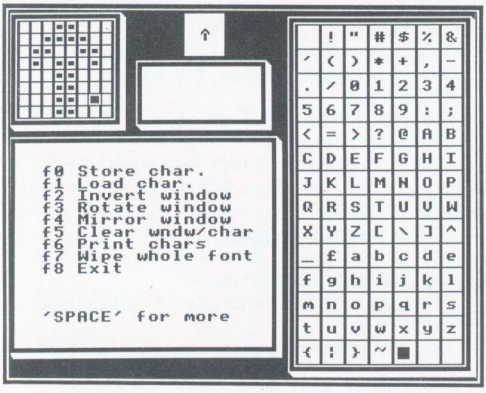


Figure 1. Font designer screen

Two programs are listed here. The first of these, called SetUp, creates two machine code routines used by the second program, and must always be run first. When its task is complete, it automatically chains the second program. This is the font designer proper, of which just part 1 is listed this month.

Type in both programs, keeping strictly to the line numbering given, and save respectively as *SetUp* and *Design*. Note that line 180 of Design has variables which establish the disc filing system to be used (DFS or ADFS), the disc drive number *(dr%)* and default directory *(dir$)*. Set these as appropriate for your system. As listed, the program assumes DFS (DISC), drive 0, directory '$'.

When you run SetUp, and in turn Design, the screen will eventually be displayed as shown in figure 1. The top left-hand window is used for editing an individual character, while the large window to the right is used to store all the characters in a font. Design is controlled by the function keys (and the cursor keys), and the purpose of each is shown in a further window at bottom left. At present, most of the functions are inoperative (to be added next month), but you can design and edit a character (use Ctrl to insert or delete a pixel), and a character design can be moved to the font display and vice versa. Note that f8 exits from the program.

That is all we have space for this time. Part 2 of the Design program will follow next month, with a full explanation of all its features, and details of how to include user-defined fonts in your own programs.

Listing 1

```
 10 REM Program SetUp
 20 REM Version B1.0
 30 REM Author  Ian Stewart
 40 REM BEEBUG  Aug/Sept 1989
 50 REM Program subject to copyright
 60 :
100 *FX225
110 *FX226
120 *FX227
130 *FX4,1
140 PROCbox:PROCprint
```

```
 150 PAGE=&1800:CHAIN "Design"
 160 END
 170 :
1000 DEF PROCbox
1010 p1=&70:p2=&72:p3=&74:p4=&76
1020 FOR F%=0 TO 2 STEP2
1030 P%=&900:[OPT F%
1040 JSR win0:JSR subp1
1050 LDA #16:JSR subp2
1060 LDA #16:JSR addp3
1070 JSR addp4:JSR win0:LDA #3:STA &80
1080 .lp1
1090 JSR win1:JSR addp1
1100 LDA #4:JSR subp2:LDA #4:JSR addp3
1110 JSR subp4:DEC &80:BPL lp1
1120 RTS
1130 .win0
1140 JSR win1
1150 LDA #5:LDX #0:LDY #6:JSR draw
1160 LDA #5:LDX #0:LDY #2:JSR draw
1170 RTS
1180 .win1
1190 LDA #4:LDX #0:LDY #2:JSR draw
1200 LDA #5:LDX #4:LDY #2:JSR draw
1210 LDA #5:LDX #4:LDY #6:JSR draw
1220 RTS
1230 .draw
1240 STA ty
1250 LDA p1,X:STA x:LDA p1+1,X:STA x+1
1260 LDA p1,Y:STA y:LDA p1+1,Y:STA y+1
1270 LDX #0
1280 .lp0
1290 LDA plot,X:JSR &FFEE
1300 INX:CPX #6:BNE lp0
1310 RTS
1320 .addp1
1330 LDA #4:CLC:ADC p1:STA p1
1340 LDA p1+1:ADC #0:STA p1+1
1350 RTS
1360 .subp1
1370 LDA p1:SEC:SBC #16:STA p1
1380 LDA p1+1:SBC #0:STA p1+1
1390 RTS
1400 .subp2
1410 STA &81
1420 LDA p2:SEC:SBC &81:STA p2
1430 LDA p2+1:SBC #0:STA p2+1
1440 RTS
1450 .addp3
1460 CLC:ADC p3:STA p3
1470 LDA p3+1:ADC #0:STA p3+1
1480 RTS
1490 .addp4
1500 LDA #16:CLC:ADC p4:STA p4
1510 LDA p4+1:ADC #0:STA p4+1
1520 RTS
1530 .subp4
1540 LDA p4:SEC:SBC #4:STA p4
1550 LDA p4+1:SBC #0:STA p4+1
1560 RTS
1570 .plot:EQUB 25
1580 .ty:EQUB 5
1590 .x:EQUW0
1600 .y:EQUW 0
1610 ]:NEXT
1620 ENDPROC
1630 :
1640 DEF PROCprint
1650 FOR F%=0 TO 2 STEP 2
1660 P%=&A00:[OPT F%
1670 STA char:STA &80:LDA #0:STA &72
1680 LDA #12:STA &73:LDA #0:LDX #&11
1690 DEX:STX &71:STA &70:LDX char
1700 .lp0
1710 LDA &70:CLC:ADC #8:STA &70
1720 LDA &71:ADC #0:STA &71
1730 DEX:BNE lp0
1740 LDA #2:JSR &FFEE:LDX #0
1750 .lp1
1760 LDA prncds,X:JSR prn:INX
1770 CMP #0:BNE lp1:LDY #0:STY &82
1780 .lp3
1790 LDA (&70),Y:STA &80
1800 LDY #0:LDA #&80:STA &81
1810 .lp4
1820 LDA &81:BIT &80:BNE set
1830 LDA #0
1840 .cont
1850 STA (&72),Y:INY:LSR &81:BNE lp4
1860 LDA &72:CLC:ADC #8:STA &72
1870 LDA &73:ADC #0:STA &73
1880 INC &82:LDY &82:CPY #8:BNE lp3
1890 JSR sub:LDY #0
1900 .lp5
1910 LDA #0:STA &80:LDA #&80:STA &81
1920 LDX #7
1930 .lp2
1940 LDA (&72),Y:BEQ noadd
1950 LDA &80:CLC:ADC &81:STA &80
1960 .noadd
1970 LSR &81
1980 LDA &72:CLC:ADC #8:STA &72
1990 LDA &73:ADC #0:STA &73
2000 DEX:BPL lp2
```

```
 2010 LDA &80:JSR prn:JSR sub
 2020 INY:CPY #8:BNE lp5
 2030 LDA #3:JMP &FFEE
 2040 .set
 2050 LDA #255:JMP cont
 2060 .sub
 2070 LDA &72:SEC:SBC #&40:STA &72
 2080 LDA &73:SBC #0:STA &73
 2090 RTS
 2100 .prn
 2110 PHA:LDA #1:JSR &FFEE
 2120 PLA:JMP &FFEE
 2130 :
 2140 .prncds
 2150 EQUB 27:EQUB 42:EQUB 5:EQUB 8
 2160 BRK
 2170 .char
 2180 BRK
 2190 ]NEXT
 2200 ENDPROC
```

Listing 2

```
   10 REM Program Design
   20 REM Version B3.1A
   30 REM Author Ian Stewart
   40 REM BEEBUG August/September 1989
   50 REM Progtam subject to copyright
   60 :
   70 *|" *** INITIALIZE *** "
   80 :
   90 MODE4:*FX154,8
  100 A%=0:Y%=0:U%=USR!&214 AND &FF
  110 IF U%=4 OR 8 THEN op%=TRUE ELSE IF
 U%=1 OR U%=2 THEN op%=FALSE ELSE VDU12,
7:PRINT"Please select DISC/ADFS or TAPE"
'"and re-run program":END
  120 PROCescF:ON ERROR GOTO 6440
  130 bbc%=INKEY-256:a$=CHR$28:b$=CHR$12
  140 IFbbc%=-1 THEN ?&367=15 ELSE *FX25
  150 w0$=a$+CHR$3+CHR$27+CHR$21+b$+b$
  160 w1$=a$+CHR$1+CHR$27+CHR$20+b$+b$
  170 w2$=a$+CHR$1+CHR$27+CHR$21+b$+b$
  180 @%=0:i%=TRUE:disc$="ADFS":dr%=0:di
r$="$":box%=&900:prn%=&A00:font%=&1100:c
opy%=&1410
  190 DIM ws1% &100,ws2% &100,pa% &100
  200 DIM gr%(8,8),val%(7):x%=1:y%=1
  210 SOUND3,-14,200,4
  220 FOR F%=font% TO font%+780 STEP 4:!
F%=0:NEXT
  230 IF disc$="DISC" OSCLI("DIR "+dir$
):PROCos("DRIVE "+STR$(dr%))
  240 IF disc$="ADFS" OSCLI("DIR :"+STR
$(dr%)+"."+dir$)
  250 VDU19,1,0;0;17,129,12,17,128
  260 FOR F%=1 TO 8
  270 FOR N%=1 TO 8:M%=FNaddr1(F%,N%)
  280 !M%=&818181FF:!(M%+4)=&81818181
  290 NEXT N%,F%:GCOL0,0
  300 PROCwndw(24,731,292,995)
  310 PROCwndw(780,76,1236,983)
  320 VDU24,784;80;1224;975;16,18,0,1
  330 FOR N%=80 TO 975 STEP 64
  340 MOVE 784,N%:DRAW 1227,N%:NEXT
  350 FOR F%=784 TO 1200 STEP 64
  360 MOVE F%,80:DRAW F%,80:DRAW F%,975
  370 MOVE F%,975:MOVE F%+4,80:DRAW F%+4
,975
  380 NEXT:PROCfont
  390 VDU24,24;128;716;667;16,26,18;0;
  400 PROCwndw(24,128,716,667):PROCinstr
(i%)
  410 VDU24,484;899;588;1007;16,26
  420 PROCwndw(368,731,688,867)
  430 VDU24,372;731;684;867;16,26,20,18,
0,1
  440 SOUND3,-14,200,4
  450 :
  460 *|" *** MAIN ROUTINE *** "
  470 :
  480 REPEAT
  490 PROCcur(TRUE,FALSE)
  500 sh%=INKEY-1:B%=gr%(x%,y%)
  510 IF INKEY-99 i%=NOTi%:PROCinstr(i%)
  520 IF INKEY-26 PROClt(FALSE)
  530 IF INKEY-122 PROCrt(FALSE)
  540 IF INKEY-58 PROCup(FALSE)
  550 IF INKEY-42 PROCdn(FALSE)
  560 IF INKEY-2 PROCbit(B%):GOTO510
  570 option%=FNkey
  580 IF option%=0 OR option%>2 THEN 710
  590 IF sh% THEN 660
  600 PROCinstr(TRUE)
  610 M%=FNaddr1(option%+12,3)
  620 FOR F%=M% TO M%+15 STEP 4
  630 !F%=!F% EOR-1:NEXT
  640 ON option% GOSUB 770,880,1020,1240
,1510,1770,1930,2550
  650 GOTO 710
  660 PROCinstr(FALSE)
  670 M%=FNaddr1(option%+13,3)
  680 FOR F%=M% TO M%+15 STEP 4
```

Continued on page 52

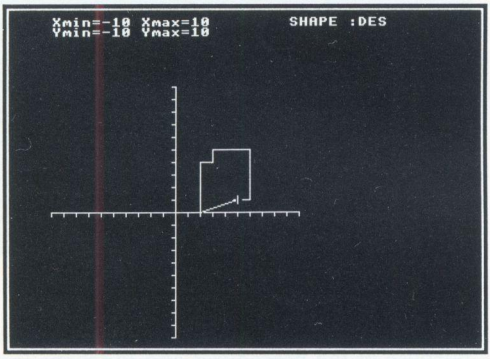
Mathematical Transformations (Part 2)

Keith Sumner adds the remaining features to this program, and describes the implementation of mathematical transformations in Basic

The new code listed this month should be added to last month's program, but do make sure that you have not changed any of the line numbers. Save the new program, preferably under a new name. The new code adds a number of important additional options in the choice of shape and transformation.

NEW SHAPES

As well as the rectangle and triangle shapes included previously, we can now define our own shape (with up to 10 points) by answering with 'DESIGN' in response to the initial prompt for a shape (instead of RECTANGLE or TRIANGLE).

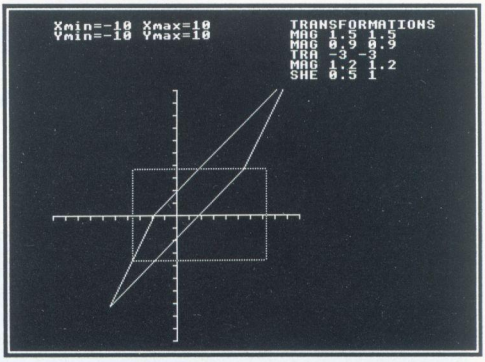


Using the shape definer

To design your own shape, use the cursor keys to move a cross hair cursor around the axes. Pressing the spacebar defines a point at the current cursor position. Use 'D' at any time to delete the shape defined so far, 'E' or Return to exit when the shape is complete.

ADDITIONAL TRANSFORMATIONS

The following transformations are now added to those described last month. Note that the abbreviated command word for those instructions which have one is shown in brackets after the extended instruction name.



Performing a shear

1. SHEAR(SHE): S(shx,shy) - This causes the object to be sheared. The matrix for this operation is:

$$(x'\ y'\ 1) = (x\ y\ 1) \cdot \begin{pmatrix} 1 & shy & 0 \\ shx & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The net effect is:

$x' = x + y.shx$
$y' = y + x.shy$

This command is executed by typing the word SHEAR followed by two parameters giving the required shear factor in first the x direction and then the y direction. The SHEAR command can be abbreviated to SHE. Thus the command format could be:

```
SHEAR 0.5 1 <Return>
```

or at its shortest:

```
SHE 0.5 1 <Return>
```

2. INVERT (INV) - The object is inverted through the origin.

3. REFLX=Y (RX=Y) - The object is reflected in the line x=y.

4. REFLX=-Y (RX=-Y) - The object is reflected in the line x=-y.

5. REFLX=0 (RX=0) - The object is reflected in the line x=0.

6. REFLY=0 (RY=0) - The object is reflected in the line y=0.

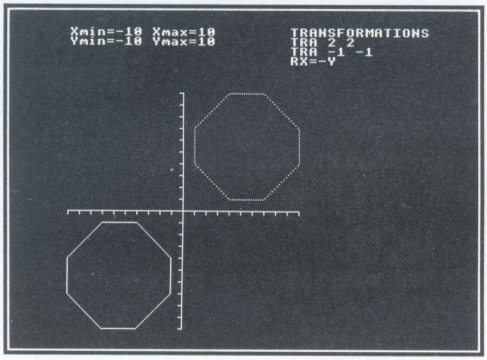
There are also some further commands which may be used here.

7. AXES - Refreshes the co-ordinate axes. This is useful if a shape lying on the axes is erased, blanking one or both axes at the same time.

8. SHOW - shows the current state of the transformation matrix.

9. CRDS - displays the co-ordinates of the vertices of the object being transformed.

10. HELP - displays a list of the commands available to the user.



A reflection in X=-Y

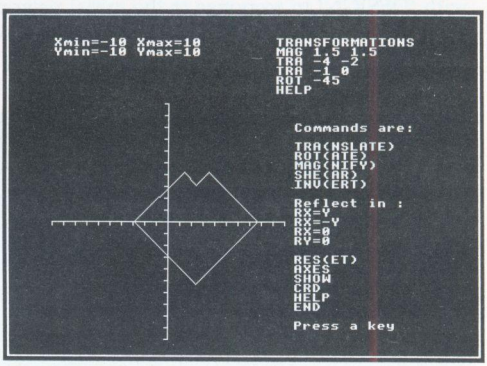
When a transformation takes place, the timing of the various steps as seen on the screen is controlled by the procedure PROCrastinate (defined in part 1 of the program at line 4370). Modify this if you wish to speed up or slow down this part of the display.

Taken together, the full range of features provided by combining parts one and two of this program provides a comprehensive environment for the exploration of mathematical transformations.

PROGRAM NOTES

The transformation routines need not be used only in the context of this program, but can be lifted to form the basis of another application. A careful study of the program code will reveal that there are in fact two different transformation matrices active at any one time. The first controls the transformation operations being performed on the selected geometrical object and is in a space mathematically local to the program. The second controls the transformations which map the object from its local space to the BBC micro's screen co-ordinate space. This implementation thus shows the power of keeping different co-ordinate transformations separate, and only applying the final composite transformation to the co-ordinates just before plotting.

Thus the transformations being performed on the object in the program's local space are held in matrix 1. The conversion of these co-ordinates to the BBC micro's screen space is controlled by the contents of matrix 0. The composite transformation is held in matrix 2 and is applied to the co-ordinates of the object just prior to plotting.



Showing the available commands

The following is an outline of the name and purpose of the procedures and functions that make up the graphics transformation kernel (lines 2500 to 3910).

PROCEDURES

PROCsetup_matrix
Defines the number of transformation matrices in the application, and also the maximum number of points transformable.

PROCset_matrix(M,A,B,C,D,E,F)
Sets the components a to f of matrix number M to the values A to F (see last month's article for an explanation of the lettering used).

PROCinitialize(M)
Sets the matrix M to be the identity matrix I.

PROCtrans(M,tx,ty,set)
Depending on the value of the flag set one of two actions occurs (its function is the same in all other procedures where it appears):

If set=TRUE the matrix is initialized and then the translation transformation is applied.

If set=FALSE the translation operation is applied to the transformation matrix in its current state thus allowing concatenation of transformations.

PROCscale(M,mx,my,set)
Allows the scaling of an object to be altered in both the x and y directions by different amounts.

PROCrotate(M,theta,set)
Enables an object to be rotated about the origin by theta degrees in an anticlockwise direction.

PROCshear(M,shx,shy,set)
Allows an object to undergo a shearing operation.

PROCmultmat(T1,T2,R)
Multiplies the matrices T1 and T2 together to produce a composite transformation matrix R, thus:
R = T1.T2

PROCmmi(M,A,B,C,D,E,F)
Multiplies matrix M with the matrix defined by the matrix components given in the call argument list.

PROCtransform(M,N)
Applies the transformation matrix M to the N points held in the arrays X() and Y(). The transformed points are placed in arrays XT() and YT().

PROCerase(P%,N)
Erases the N points held in XTMP() and YTMP() from the screen.

PROCdraw(P%,N)
Draws the N points held in XT() and YT() on the screen.

PROCstore_mat(M,N)
Stores the contents of matrix M in matrix N.

PROCload_mat(M,N)
Loads the matrix M with contents of matrix N.

PROCreflectx(M,set)
Enables the object to be reflected in the line x=0.

PROCreflecty(M,set)
Enables the object to be reflected in the line y=0.

PROCinvert(M,set)
Enables the object to be inverted in the origin of co-ordinate space.

PROCreflectx_eq_y(M,set)
Enables the object to be reflected in the line x=y.

PROCreflectx_eq_minus_y(M,set)
Enables the object to be reflected in the line x=-y.

PROCprint_matrix(M)
Allows the contents of the 'local program space' matrix (Matrix 1 in the context of this application) to be inspected.

FUNCTIONS

FNtfmx(M,x%,y%)
This function transforms the co-ordinate pair (x%,y%) according to the matrix M. The value returned by the function is the new x co-ordinate.

FNtfmy(M,x%,y%)
This function transforms the co-ordinate pair (x%,y%) according to the matrix M. The value returned by the function is the new y co-ordinate.

```
1310 IFFNstr("D",0) PROCdesign:ok=nc>0
1550 DEF PROCdesign
1560 xst=2:yst=2:*fx4,1
1570 nc=0:X(nc)=xst:Y(nc)=yst
1580 PROCtransform(0,nc+1)
1590 PROCcursor
1600 REPEAT:*fx15,0
1610 A=GET
1620 IF A=136 ANDxst>-10 PROCupd(-1,0)
1630 IF A=137 ANDxst<10 PROCupd(1,0)
1640 IF A=138 ANDyst>-10 PROCupd(0,-1)
1650 IF A=139 ANDyst<10 PROCupd(0,1)
1660 IF A=32 PROCadd
1670 IF A=68 PROCdel
1680 UNTIL(A=69 OR A=13) OR nc=npts
1690 PROCcursor:*fx4,0
1700 PROCaxes(TRUE):PROCdraw(4,nc)
1710 ENDPROC
1720 :
1730 DEF PROCdel
1740 PROCaxes(TRUE)
1750 nc=0:xst=X(nc):yst=Y(nc)
1760 PROCtransform(0,nc+1)
```

```
 1770 PROCcursor
 1780 ENDPROC
 1790 :
 1800 DEF PROCadd
 1810 PROCcursor:PROCdraw(13,nc)
 1820 nc=nc+1:X(nc)=xst:Y(nc)=yst
 1830 PROCtransform(0,nc+1)
 1840 PROCdraw(12,nc):PROCcursor
 1850 ENDPROC
 1860 :
 1870 DEF PROCupd(dx,dy)
 1880 xst=xst+dx:yst=yst+dy
 1890 X(nc)=xst:Y(nc)=yst
 1900 PROCcursor:PROCtransform(0,nc+1):P
ROCcursor
 1910 ENDPROC
 1920 :
 1930 DEF PROCcursor
 1940 GCOL3,1
 1950 MOVE XT(nc)-tick,YT(nc):DRAW XT(nc
)+tick,YT(nc)
 1960 MOVE XT(nc),YT(nc)-tick:DRAW XT(nc
),YT(nc)+tick
 1970 ENDPROC
 1980 :
 2010 PROChelp
 2070 IF FNstr("INV",0) PROCexec(3)
 2080 IF FNstr("RX=0",0) PROCexec(4)
 2090 IF FNstr("RY=0",0) PROCexec(5)
 2100 IF FNstr("RX=Y",0) PROCexec(6)
 2110 IF FNstr("RX=-Y",0) PROCexec(7)
 2120 IF FNstr("SHE",2) PROCexec(8)
 2140 IF FNstr("AXES",0) PROCaxes(TRUE):
PROCdraw(4,nc)
 2150 IF FNstr("SHOW",0) PROCprint_matri
x(1)
 2160 IF FNstr("CRD",0) PROCprint_coords
 2170 IF FNstr("HELP",0) PROChelp
 3610 DEF PROCreflectx(M,set)
 3620 IF set PROCset_matrix(M,1,0,0,0,-1
,0) ELSE PROCmmi(M,1,0,0,0,-1,0)
 3630 ENDPROC
 3640 :
 3650 DEF PROCreflecty(M,set)
 3660 IF set PROCset_matrix(M,-1,0,0,0,1
,0) ELSE PROCmmi(M,-1,0,0,0,1,0)
 3670 ENDPROC
 3680 :
 3690 DEF PROCinvert(M,set)
 3700 IF set PROCset_matrix(M,-1,0,0,0,-
1,0) ELSE PROCmmi(M,-1,0,0,0,-1,0)
 3710 ENDPROC
 3720 :
 3730 DEF PROCreflectx_eq_y(M,set)
 3740 IF set PROCset_matrix(M,0,1,0,1,0,
0) ELSE PROCmmi(M,0,1,0,1,0,0)
 3750 ENDPROC
 3760 :
 3770 DEF PROCreflectx_eq_minus_y(M,set)
 3780 IF set PROCset_matrix(M,0,-1,0,-1,
0,0) ELSE PROCmmi(M,0,-1,0,-1,0,0)
 3790 ENDPROC
 3800 :
 3810 DEF PROCprint_matrix(M)
 3820 VDU28,22,26,38,20:@%=&20204
 3830 FOR R%=1 TO 3:FOR C%=1 TO 3
 3840 PRINTTAB((C%-1)*6-(mat(C%,R%,M)>=0
),R%);mat(C%,R%,M);
 3850 NEXT:PRINT:NEXT
 3860 PRINT'"Press a key";:*FX15,0
 3870 A=GET:CLS:@%=10
 3880 PROCaxes(FALSE):PROCdraw(4,nc)
 3890 ENDPROC
 3900 :
 3930 DEF PROCprint_coords
 3940 VDU28,26,28,38,12:@%=&20206
 3950 FOR N=0 TO nc-1
 3960 PRINT;FNtfmx(1,X(N),Y(N)),FNtfmy(1
,X(N),Y(N)):NEXT
 3970 PRINT'"Press a key";:*FX15,0
 3980 A=GET:CLS:@%=10
 3990 ENDPROC
 4000 :
 4060 IF func=3 PROCinvert(1,FALSE)
 4070 IF func=4 PROCreflecty(1,FALSE)
 4080 IF func=5 PROCreflectx(1,FALSE)
 4090 IF func=6 PROCreflectx_eq_y(1,FALS
E)
 4100 IF func=7 PROCreflectx_eq_minus_y(
1,FALSE)
 4110 IF func=8 PROCshear(1,ta(1),ta(2),
FALSE)
 4260 DEF PROChelp
 4270 VDU28,26,30,38,9
 4280 PRINT"Commands are:"
 4290 PRINT"TRA(NSLATE)"'"ROT(ATE)"'"MAG
(NIFY)"
 4300 PRINT"SHE(AR)"'"INV(ERT)"'"Reflec
t in :"
 4310 PRINT"RX=Y"'"RX=-Y"'"RX=0"'"RY=0"
 4320 PRINT'"RES(ET)"'"AXES"'"SHOW"'"CRD
"'"HELP"'"END"
 4330 PRINT'"Press a key";:*FX15,0
 4340 A=GET:CLS
 4350 ENDPROC
 4360 :
```

Introducing Postscript

David Spencer takes a look at the language of laser printers.

In BEEBUG Vol.7 No.10 we took a look at laser printers, and mentioned the Page Description Language (PDL) *PostScript*. Now, in this short series we will take a more detailed look at *PostScript*. You might wonder why BEEBUG, a BBC-orientated magazine, should feature articles on a very specialised programming language which few members will have access to. However, there are two main reasons for this. Firstly, *PostScript* is becoming increasingly common, both in terms of the number of different systems supporting it, and the numbers of actual systems in use. In particular, many schools are now using Apple Macintoshes connected to LaserWriters - one of the most common *PostScript* printers. In the future, the use of *PostScript* can only become more widespread, especially with the current trend towards '*Screen PostScript*'. With this, the operating system of the computer features a *PostScript* interpreter which generates its output on screen rather than on paper. All applications then perform their screen output by sending *PostScript* commands to this interpreter. This simplifies application programs because a common *PostScript* driver can be used for both the screen display and the printed output, and it also ensures that the screen display produced by a program will match as closely as possible the final printed output. Therefore, all programs are inherently WYSIWYG (What You See Is What You Get). The second reason for studying *PostScript* is that in terms of a programming language it is very different from the likes of Basic and C, and is well worth attention in its own right.

A HISTORY LESSON

PostScript is the brainchild of a handful of American computers programmers who formed a company called Adobe Systems in 1982. They recognised that with the increasing sophistication of printers, and the software driving them (particularly on the Apple Macintosh), there was a desperate need to develop a standard that would allow any piece of software to drive any printer. Furthermore, they realised that something along the lines of the Escape sequences used to control dot-matrix printers was not going to be sufficient to cope with the needs of future software. Therefore, what Adobe did was develop an entire programming language which could be used to describe the layout of text and graphics on a page, and hence *PostScript* was born.

THE IMAGE MODEL

As *PostScript* is designed to specify a page layout, a good starting point in its study is to consider how the language refers to the page. One of the most important features of *PostScript* is that it is almost totally device independent from the user's point of view. This ensures that a *PostScript* program written for a 300 dpi laser printer will produce the correct results if run on a photo-typesetter working at 4500 dpi. What's more, the program running on the typesetter will take full advantage of the extra resolution offered by that machine. To achieve this device independence, a *PostScript* program refers to positions on the page using a co-ordinate system known as User Space. By default, User Space has its origin at the bottom left corner of the paper, with the X-axis horizontal, and the Y-axis vertical. Measurements are in units of 1 point, which corresponds to 1/72". (The *point* is a traditional printing measurement which has been adopted in Desktop Publishing.) For example, an A4 sheet of paper is 11.69 inches high and 8.27 inches wide, which means that the coordinates of the top right hand corner in points are (595,842). As we will see shortly, the User Space coordinate system can be transformed in any way possible, so you could for example have the origin at the top of the page, or in the centre, or you could scale the coordinates to work in inches or millimetres instead of points.

When it comes to printing marks on the paper, the ink colour can be specified as black, or as any colour using one of two methods for specifying colour. Obviously colour printers are a rarity, and not many *PostScript* devices will be able to reproduce colour correctly. However, the nature of *PostScript* means that even on a mono printer, coloured areas will be reproduced in the best possible way. Whatever the colour of the ink, it is considered as being opaque which means that any marks printed on the paper will totally overwrite anything already there. Hence, effects such as Oring which can be used on-screen cannot be reproduced on a *PostScript* printer.

TRANSFORMATION MATRICES

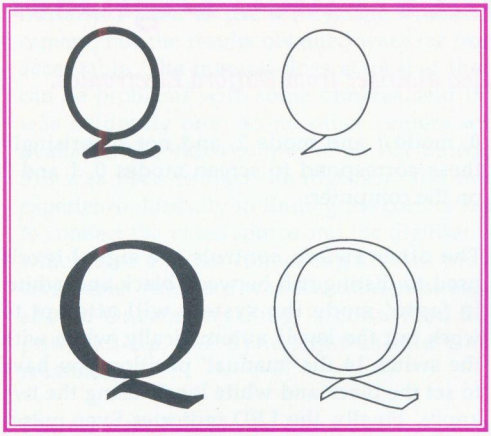
Another key feature of *PostScript* is the concept of a transformation matrix. The basic idea is that by using a three by three matrix, any coordinate can be translated and subjected to any combination of affine transformations (such as scaling, rotating and shearing) simply by multiplying the coordinate by the matrix. By multiplying a number of transformation matrices together a single matrix is formed which represents all the component transformations. For more details of this see the Transformations program featured in this, and the previous issue of BEEBUG.

PostScript maintains a Current Transformation Matrix (CTM) which is used to transform all coordinates used within a program. Initially, the CTM maps coordinates from the User Space into the so-called Device Space. This is the coordinate system that is native to the physical output device. Typically, one unit in the Device Space corresponds to one pixel on the printed page. Commands exist within *PostScript* to multiply the CTM by another matrix, thereby adding additional transformations to those used to map coordinates onto the physical output device. It is by modifying the CTM that effects such as moving the origin, or scaling coordinate units can be achieved. As all coordinates are scaled by the CTM, this is a very powerful feature. For example, if the CTM is modified so as to rotate the axes by 90 degrees, then anything subsequently printed, including text, will be rotated. Similarly, if a PostScript procedure is written to draw a square with sides of one inch starting at the origin, then by modifying the CTM the same procedure can draw any sized square at any place and any orientation.

PATHS, CHARACTERS AND IMAGES

Printed objects in *PostScript* can be in one of three forms, Paths, Characters and Images. Perhaps the most important type of objects are characters. These can be in a variety of type styles (fonts) and at any size. It is even possible to have the font size different in the X and Y directions, allowing horizontally scaled characters to be printed. Other character effects can also be achieved, such as printing characters as outlines, or 'shadowing' characters. The fonts available on a *PostScript* printer depend on the particular device, and are not defined as part of the language. The most commonly used fonts will be stored in ROM within the printer, while some devices allow further fonts to be stored on a hard disc drive built into the printer. Obviously features like this are only to be found on the more up-market *PostScript* systems such as photo-typesetters. Another way in which the printer can access fonts is by them being downloaded from the host computer into RAM within the printer. In this way, a number of fonts can be stored by the host computer, and only downloaded when needed. This can be done intelligently, for example, the Apple Macintosh examines any *PostScript* programs that are sent to the printer, and if they require fonts which don't exist in the printer, it attempts to download the font. Whatever the source of the font, it will normally exist in a number of styles, for example plain, bold, italic and bold italic.

Paths are a sequence of imaginary lines which are drawn on the paper. These lines can be straight, or can be a special type of curve known as a Bezier curve. The theory of these is too involved to explain here, but they are basically curves which are specified in terms of their endpoints, and two intermediate 'control



A character from Courier (top) and Palatino showing the difference between thin-stroke and outline fonts

points' which affect the direction and curvature of the line. A path can also be closed, which means that its end is linked to the start. (There is a subtle difference between this and the start and end points happening to be in the same position, as we shall see next month.) The imaginary lines which make up a path are of infinitesimal thickness, and therefore don't in themselves produce any marks on the paper. However, commands exist within *PostScript* to Stroke a path, which means broadening out the path to make actual lines and, in the case of a closed path, to fill its interior with solid colour. These operations are very flexible. For example, when stroking a path you can specify a solid line, or any pattern of dotted or dashed lines. You can also specify how the join between lines is to appear, and how the ends of the path (if it is not closed) are to be finished off. Other operations on paths are also possible, for example you can make the current path a clipping path, which means that subsequent marks are only placed on the page if they fall within the path. This is similar to the concept of a graphics window on the Beeb, but much more flexible because the path can be of arbitrary shape.

The final type of object is the image. This is merely a pixel by pixel representation of a rectangular area, and is intended primarily for tasks such as including scanned photographs in a document. The image can be any size, and can of course be transformed by the CTM, just as with any other object.

In fact, the distinction between characters and paths is artificial, because *PostScript* stores characters internally as paths, in one of two forms. Most fonts are stored as outlines, which means that the path for each character represents the outline of the character. Certain thin fonts (such as Courier - a typewriter style font) are instead stored as a path which corresponds to the lines that make up the character. To make this clearer, figure 1 shows a 'Q' in both Courier and Palatino (a very common serifed font), together with the paths that make up the character. When a particular font is selected the language sets up a mapping between the ASCII codes of each character in the font, and the path which represents that character. When a font is scaled to a particular size what in fact happens is that the path for each character is scaled by the appropriate value. When a character is printed, the components of its path are added to the current path. Then, the action taken depends on the type of character. For an outline, the path is filled, while for a thin font it is stroked with a line thickness appropriate to the font size.

Having looked at how a *PostScript* program views the printed page (or in fact whatever output device the interpreter is driving), we will move on next month to study the language itself, and develop some simple programs. We will also give details of how to connect a Beeb to an Apple LaserWriter Plus, perhaps the most common *PostScript* printer. Meanwhile, if you want to find out more about *PostScript*, then there are three books all published by Adison Wesley and Adobe Systems. These are *'PostScript - Tutorial and Cookbook'*, *'Postscript - Reference Manual'*, and *'PostScript - Program Design'*. The Cookbook costs £14.95, and the other two £19.95 each, and they should be available from any bookshop with a computer science section.

The Watford Electronics Video Digitiser

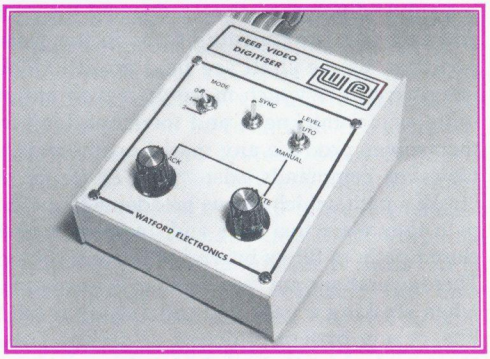
David Spencer takes a look at the Beeb Video digitiser from Watford Electronics.

Product	**Beeb Video Digitiser**
Supplier	**Watford Electronics, Jessa House, 250 Lower High Street, Watford, Herts WD1 2AN. Tel. (0923) 37774**
Price	**£130.35 (inc. VAT and p&p)**

The idea of a video digitiser is a simple one - it is a device for taking a picture from some video source, and converting it into a series of discrete pixels which can be displayed by the computer. However, in practice the situation is more involved. Firstly, the sampling of the picture needs to be done quickly, which necessitates a good deal of electronics, and secondly, the sampling must be coordinated with the sync pulses produced by the video source, otherwise the picture will be jumbled up randomly, and will appear as nonsense. These reasons, together with the limited screen resolution offered by the Beeb have meant that video digitisers for the Beeb have been almost non-existent. However, there is one, produced by Watford Electronics, which has been available for some time now, and it is this which we will look at here.

The digitiser comes in the form of a grey plastic box about six inches by four inches, with a sloping front that is two inches high at the back, and one inch at the front. For some reason, having seen the picture of the device in Watford's advert, I expected it to be about twice the size that it actually is. Protruding from the back of the unit is a 20-way ribbon cable that connects to the computer's user port, and which is about a metre and a half long. Also on the back is a BNC-type socket for connecting the video source (more of which later). On the front of the case are two switches, two knobs, and an LED. The left hand switch has three positions marked mode 0, mode 1 and mode 2, and not surprisingly these correspond to screen modes 0, 1 and 2 on the computer.

The other switch controls the signal levels used to distinguish between black and white. In 'auto' mode the system will attempt to work out the levels automatically, while with the switch in the 'manual' position you have to set the black and white levels using the two knobs. Finally, the LED indicates Sync pulses from the video source, and should flash on and off about once every two seconds provided a video source is connected to the digitiser, and the software is installed in the computer.



The software is supplied on a 16K EPROM which will need to be installed in the computer. If you are tight on ROM sockets then I suspect that an image of the ROM could be transferred to disc and loaded into sideways RAM, though I have not tried this. Having installed the software, the next stage is to connect a suitable video source. The digitiser needs to be connected to the video-out socket on either a camera or a video recorder. The system is compatible with both 75 ohm monochrome and composite colour (PAL) signals, though the images are always grabbed in black and white.

Initially, I tried to use a Ferguson Videostar camera, but the results obtained were far from acceptable. The manual does state that there can be problems with some cameras, and this was evidently one. As no other camera was available, the video output of an Archimedes 310 was used to feed the digitiser. You may experience difficulty in finding the correct lead to connect the video source and the digitiser, in which case a video connecting kit will prove invaluable. These are available from most video shops. Once everything is connected and powered up, the Sync light should flash reassuringly.

The digitiser is controlled by a number of star commands, though these can be packed up into a more usable form, as shown by the program elsewhere in this issue. The most important command is *IMAGE which grabs a screen from the digitiser. For this to work, the computer must be in the same mode as the switch on the digitiser, and the screen must not have been scrolled. The only parameter that *IMAGE can take is an 'N' to grab a negative of the image. It takes a couple of seconds to grab a single image, and therefore when using a camera it is best to mount it on a tripod. Similarly, the picture should be frozen if a video recorder is being used.

The commands *IMPRNT and *OPRINT dump the grabbed image to an Epson or Acorn Sparkjet printer respectively. An optional aspect ratio can be specified to ensure that the dimensions of the dump match those of the original picture. One very annoying feature of these commands is that they assume that the printer is setup to produce linefeeds each time a carriage return is received. If this is not the case, which it won't be for many printers, then the entire dump is printed on a single line. You therefore have to open up the printer and change the DIP switch settings.

*IMLOAD and *IMSAVE load and save grabbed images in a compressed format. Watford claim that a 20K screen will be reduced to between 1 and 12K, depending on complexity. Finally, the commands *IMART, *MASK and *IMMIX are used to convert a grabbed image into a format suitable for transfer to the AMX art packages and Stop Press.

In use, the only problem I found with the digitiser was setting it up. The 'auto' mode proved to produce a picture which was too dark, and lacking in contrast. I therefore had to resort to using 'manual' mode, and adjusting the two threshold controls in order to obtain a suitable image. The best way to do this is to use a simple program which repeatedly grabs images. Once setup, the controls should not need adjusting unless the video source is changed. The manual suggests that the setting up is performed in mode 2, however this is only of use on a monochrome monitor, as the colours prove to make the image almost unrecognisable. However, with a monochrome system, the colours in mode 2 are used to good effect to provide grey shades.

The documentation is in the form of a twenty-four page manual, with an additional section covering ROM installation. The manual explains what a digitiser is, gives details of all the available commands, and describes how to set up the system. There are also sections on using the commands within your own programs, and fault finding.

CONCLUSION

Having found a suitable video source, the digitiser proved to perform very well, and the results produced are probably as good as is possible on the Beeb. My only niggle is that the printer dumps assume a particular printer setting, but apart from that I cannot fault the system. However, I would advise anyone who is considering buying the system to check if it is compatible with the camera they intend to use. The price of the unit also seems reasonable for what it offers.

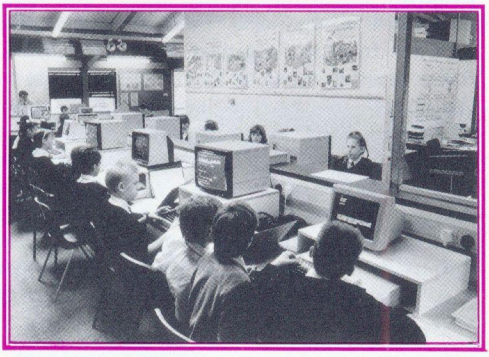
Coping with Computers in Schools

At the start of a new school year, Paul Pibworth, an experienced teacher and head of department in a comprehensive school, presents a personal view on the way schools and teachers are coping with information technology.

Following the micro revolution, computers became an accepted part of school equipment. With the introduction of the National Curriculum, and the inclusion of Information Technology (IT), their presence has become a necessity. Many homes now have a micro-computer, although in many cases it will not be the same model as the one in the child's school. Education pages feature regularly in the computer press. They often take the form of surveys of the current software scene, and are aimed at both teachers and parents. This article is not another software review, but an attempt to explore some of the difficulties which can arise as schools move forward.

It may well be helpful to survey briefly the use of computers in schools. I am aware that brevity may cause over-simplification, but it will help to set the scene. Compulsory state education tends to be divided into one of two systems: primary/secondary, and primary/middle/secondary. The younger children tend to spend most of their time in the same room with the same teacher, while at the secondary level, it is common for teaching to be subject based, and often room based. This will obviously affect the learning strategies used. Thus, a micro-computer based in a primary school classroom could be available all day for the members of that class. A computer based in a science laboratory will tend to be available to members of a given class only when that class is in that laboratory.

Another difference is the type of use to which the computer is put. Software can be content based. As an example, a program which deals with chemical formulae and equations is of little use outside a chemical laboratory. There is a wide range of such software, as a glance at any review of the type mentioned above will demonstrate. However, the in-words now are "content-free" and "cross-curricula". This approach sees the computer as a general tool, a device to help in the development of broader skills. Such skills may be classified as numeracy, communication, problem-solving, manipulation, etc. Software in this category would include wordprocessors, databases, spreadsheets, data-logging, and CAD. Finally, there is general administration software, used by teachers and the school office rather than by pupils.



With such a wide divergence of users and usage, I want to examine the situation from different angles, namely software, hardware, and people. Since the first two are often related, some overlap cannot be avoided.

SOFTWARE

I will deal first with content-based software. The first and obvious problem for the teacher is whether the software is good. The true value of software, like a textbook, becomes more apparent in use, rather than by inspection. Unlike textbooks, software is not usually made available for inspection for obvious reasons. Some LEAs are now purchasing software to be kept at a central location, but this makes the assumption

that teachers are willing to visit the centre in their own time (and probably at their own expense), a very sensitive issue at the moment!

Having decided that the software is good, the next question is whether it is suitable for the class situation. How many children can be grouped round one monitor? Who "drives" the machine? Or, given individual application, can the lesson be structured appropriately so that individual pupils can all have their turn? This is less of a problem at primary level, but could be quite difficult in a history lesson at secondary level.

Much progress has been made with content-free software. I well remember low ability pupils having difficulties with Wordstar, the philosophy then being that even low ability pupils should have access to a sophisticated word processor. This type of problem is reduced if the word processor is easy to use. I know a Modern Language department which has greatly benefitted from the purchase of Folio, a specialised word processor for use with foreign languages. On the whole, I think that great progress has been made in the area of software, and that problems in this field are becoming less significant.

HARDWARE

It is hardware that is probably the greatest source of headaches! I know that this is a BBC/Acorn users' magazine, and that BBCs feature prominently in schools. However, many 'mix-n-match' situations exist already. I know of an establishment that began with an RML380Z, added several BBCs, and then acquired an RML480Z network. The Special Needs department uses Spectrums. The library and the office have three systems between them. Finally, the school was given some Amstrad machines. There are often valid reasons behind such a situation, finance being only one. In fact, being in the vanguard can bring serious disadvantages. This is seen when those so recently lagging behind begin to to overtake as they acquire the latest 16/32 bit computers!

Then there is the problem of where to place the machines. Should there be a computer room? Individual computers are more vulnerable to theft, whereas a dedicated room is more easily fitted with an alarm system. Such a room will be less suitable for content-based software though. A teacher cannot be expected to delay a topic until the room becomes available, or to move a class if the computer component is not a significant part of the lesson. Should there be a network? A network system needs a network manager. This requires time. Teachers are paid to teach. Yet if technical help is unavailable (I speak from experience), the network still has to be administered.

The total number of machines is affected by finance. I am sure that the number of computers within schools owes much more to the fund raising of individuals and groups than to those who hold the purse strings! How many pupils still have to share a keyboard, as they undertake tasks which are part of their coursework?

Finally, while on the subject of hardware, what about maintenance? The nearest official repair depot for one of our systems is over 50 miles away! Our County technical help service is stretched because their work includes visits to primary schools to change batteries in their Masters! If a computer needs to be taken for local repairs, this again depends on a teacher going in his/her own time, and usually at his or her own expense. It is probably not widely known that some insurance companies place an

additional premium on private motor policies for this privilege, so check first!

THE HUMAN FACTOR

Turning to people, one first thinks of the pupils. Did you know that placing a magnet in front of a monitor can do permanent damage? Can you think of any reason why pupils should want to remove the fuses from the buffer boxes? When using a network, why can't pupils stay within their own user area? Why do they forget to use the establishment filename system? How did someone create a file with the name ";"...I cannot delete it!

The modern pupil is certainly not keyboard-shy; not so teachers. Perhaps the biggest obstacle to increasing the use of computers is the teachers themselves. The policy makers must bear some of the responsibility. In-service training days (Baker days as they were known) are used to discuss curricula, profiles, performance (of pupils and staff), and assessment. Should more time be given to teachers to become confident in using modern technology? In my experience, most teachers who are happy to use a computer at school do in fact have one at home! One of the rewards of teaching is the delighted response from a pupil when a new technique has been learnt, or problem has been solved. In our department, we use a system to record and process test marks. Not all staff rush to enter their marks. You can thus share my feeling of achievement when a colleague, having entered his test marks, felt brave enough to add a new pupil to a file. There was definite joy in his face as he told me, "Paul, I've added a new boy to the list". Another convert?

LOOKING AHEAD

And what about the future? How can the problems be overcome? Both the DES and the DTI have indicated support for increased computer use in schools, but how can this support be translated in practicalities?

Let us look at the 'mix and match' situation. It is with us and will stay with us. It is not an insuperable problem, but the answer is outside of the school. When software is written for a range of systems, site licences must take this into account. One is often allowed to make a back-up for use within the establishment. In this case, why not allow a second copy/version for the other system at nominal cost? What is also needed is support from the major hardware manufacturers for inter computer link-ups as is done by Cambridge Computers for its Z88. This, coupled with the necessary software to transfer data files, would be a significant step forward.

Turning to software, it is acknowledged that software houses need to pay their bills! I feel that general purpose software is more difficult to learn than that written with a particular use in mind. There is room here for LEA's to identify such applications. They could then sponsor a software house to produce the necessary software. On the other hand, there must be a significant amount of 'in-house' software already within the authority. The network administration programs that I have been using were all written by a former colleague. Such work should be identified, evaluated, and published.

Finally, I return to the teacher. It is the teacher who is in the front line, and who has to implement any new policy. It is no use putting a computer into the hand of someone who doesn't want it, doesn't like it, and who doesn't want to like it either! The battle is to convert the teacher, and commercial techniques may well need to be applied. If a business venture feels that its staff need training, they are trained in work time, at work expense, and in training centres that employ good cooks! When a drug company wishes more patients to benefit from its products, it seeks to convert the GP, over lunch of course! Some of these techniques have been used in the past, to introduce TVEI, and JIG-CAL. If school administration staff are deemed worthy of such attention, then why not school teaching staff?

As Archimedes might well have said "Convince the teacher, and you will convince the world."

1st course

Investigating Teletext Mode (5)

To conclude this short series on the use of Teletext characters and graphics, Mike Williams describes a set of useful procedures.

Having described most of the features and characteristics of Teletext mode, I propose in this final part to present a number of procedures which I have found useful. I would, however, stress one point right at the outset. My object is to encourage and help you to write your own procedures to suit your own needs. I hope therefore that you will take my examples as no more than guidelines to what can be achieved. It is not my intention that these should be seen as a definitive set of procedures which you must use. I have also included a demonstration program to show what the procedures can do and how they can be used within a program.

Why use procedures anyway? Well, if any routine is needed at different points throughout a program it makes sense to code the routine as a procedure. However, it can often be helpful to code a routine in this way even if it is called only once. It helps to keeping the programming simple, and means that the routine can be treated and tested independently of the rest of the program.

In writing procedures to work in mode 7, you need to decide on some basic parameters, and then stick to them. Most of the procedures will need to refer to a location on screen at which something is to be displayed. As previously, I prefer to make the co-ordinates refer to the visible item (text or graphics), positioning any teletext control characters to the left. The alternative, which I don't like, is to place the control characters at the co-ordinates given, but this can make alignment difficult. However, you will always need to check, with the method that I have used, that you have allowed sufficient space for any control characters (for example, you cannot use either of the first two procedures listed below to position coloured text starting in position zero on a line).

Some procedures will also need to specify a colour. The text colours range from 129 to 135, and so could be represented by the range 1 to 7, adding 128 automatically within the program. On this occasion I have decided to use colour control codes as they are, although I have used the other approach in the past.

Another factor with regard to the use of procedures is the way in which the more basic and fundamental routines can be used to build up more complex procedures, without excessive duplication of effort. However, I would once again stress the importance of adapting any routines to your own requirements.

DISPLAYING TEXT

The first two procedures are the most obvious, routines to display messages in either single or double height characters. We have covered double height characters earlier in these articles. Notice the consistency in the use of parameter and variable names to help you remember what everything is for.

```
1000 DEF PROCmsg1(x,y,c,text$)
1010 PRINTTAB(x-1,y)CHR$(c);text$;
1020 ENDPROC
1030 :
1100 DEF PROCmsg2(x,y,c,text$)
1110 LOCAL p
1120 FOR p=0 TO 1
1130 PRINTTAB(x-2,y+p)CHR$(c);CHR$141;t
ext$;
1140 NEXT p
1150 ENDPROC
1160 :
```

In both cases, x and y are the position on the screen (x being the position along the line starting from zero, and y counting the number of lines from the top of the screen downwards, again starting from zero). The parameter c is the text colour (in the range 129 to 135), and text$

holds the message to be displayed. Notice how in the first routine the single control code is placed in position x-1, while in the second routine the control codes start in position x-2 because of the extra control code for double height characters.

A variation on the second procedure would be to add a further CHR$140 at the end of the text message (line 1130) so that any further text on the same lines will be single height, not double height. However, as a result of some quirk, double height text can only be followed by single height text on the upper of the two lines in question. On the whole, it's better to keep to double height only on a particular line.

FRAMES AND BORDERS

Another procedure which I have used on several occasions, and which can make mode 7 displays look quite smart, is one for drawing rectangular frames or boxes. We need to pick the right Teletext graphics codes to form the four sides of the rectangle (which can largely be drawn with FOR-NEXT loops), but the four corner characters have to be placed individually.

A major decision needs to be made on whether the colour of the frame is to be included as one of the parameters or not. Superficially, the answer might seem obvious ('yes'), and my first version of the procedure listed below was written on this basis. However, it can often be useful to draw two or more frames adjacent to each other, but to do so will often lead to corruption of the graphics characters by the colour control codes. In the end I therefore decided to treat colour separately. The following procedure will therefore draw a frame in whatever is the current graphics colour for each line.

```
1200 DEF PROCframe(x,y,w,h)
1210 LOCAL p
1220 PRINTTAB(x,y)CHR$183;
1230 PRINT STRING$(w-2,CHR$163);
1240 VDU235
1250 FOR p=y+1 TO y+h-2
1260 PRINTTAB(x,p)CHR$181
1270 PRINTTAB(x+w-1,p)CHR$234;
1280 NEXT p
1290 PRINTTAB(x,y+h-1)CHR$245;
1300 PRINT STRING$(w-2,CHR$240);
1310 VDU250
1320 ENDPROC
1330 :
```

Of the parameters, x and y mark the position of the top left-hand corner of the frame, and w and h represent the width and height of the frame. Lines 1220 to 1240 produce the top left-hand corner, the top edge, and then the top right-hand corner. Lines 1250 to 1280 create the left and right sides of the rectangle, and lastly lines 1290 to 1310 finish off with the bottom left-hand corner, the bottom edge, and the bottom right-hand corner.

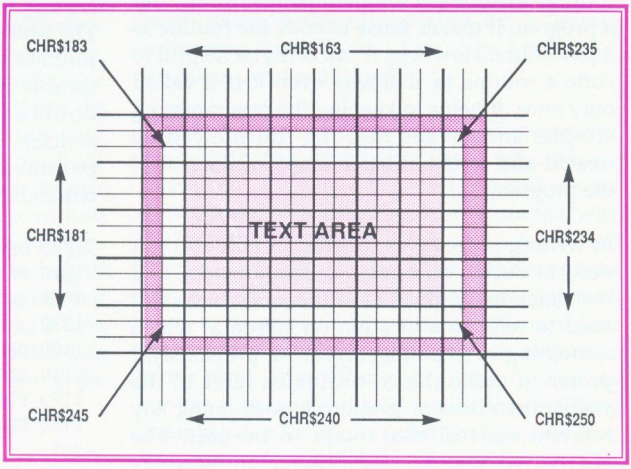


Figure 1. Typical 'outer' frame showing graphics characters used

I have chosen the graphics characters (see the back of your User Guide) to give a frame one

pixel wide (remember that a Teletext graphics character is two pixels wide by three pixels high). Several styles of frame are possible, and the one I have chosen is illustrated in figure 1. This is what I would call an outer frame. You could similarly design an inner frame. The outer frame style has the advantage that any text inside the frame is always separated by at least one pixel from the frame itself.

Be warned of one failing in this system. If you try to position double height characters, as a title say, inside a frame you will end up with a hole in the frame. This again derives from some peculiarity in the implementation of double height characters, and there is unfortunately nothing that can be done about this.

One final point to note regarding the use of the frame procedure is that the minimum size of frame is 3 vertically by 2 horizontally (not 2 by 2 as you might expect). Because FOR-NEXT loops are always executed at least once (even if this appears logically wrong in a particular application), the two vertical sides will always be at least one unit in length with the corners still to be added on.

Incidentally, if you intend to create even a modest mode 7 display, it can help to plan everything out on paper first, and the User Guide contains a handy form for just this purpose.

FOREGROUND AND BACKGROUND

Since I have decide to control graphics colours separately from the graphics itself, our next need is for a procedure which will determine a graphics colour, and while we are about we will also include a procedure to control the background colour.

A graphics colour control code needs to be placed on any line containing graphics characters. A simple arrangement is to place a column of such characters down the left-hand side of the screen to fix the graphics colour for the screen. Our procedure does just this but in a more flexible way. In effect, it allows a column of graphics colour control characters to be placed in any position, and the top and bottom of the column can also be specified.

```
1400 DEF PROCforeground(x,y1,y2,c)
1410 LOCAL p
1420 FOR p=y1 TO y2
1430 PRINTTAB(x,p)CHR$(c);
1440 NEXT p
1450 ENDPROC
1460 :
```

In this procedure, x is the position across the screen where the column is to appear, while y1 and y2 mark the positions of the top and bottom respectively of our column. If y1 and y2 are the same, then only one line will be affected.

For example, cyan is coded 150 and green 146. Thus:

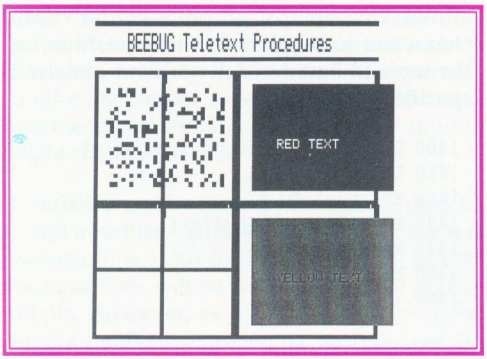
```
PROCforeground(0,0,11,150)
PROCforeground(0,12,23,146)
```

would put the code for cyan at the beginning of the first 12 lines, and the code for green at the start of the next 12 lines. All graphics characters on those lines would be in a colour determined by the control code at the start of the line. Any frames drawn with our frame drawing procedure would also be coloured by these codes.

That deals with the foreground (for graphics). Here's a procedure for colouring the background.

```
1500 DEF PROCbackground(x,y1,y2,c)
1510 LOCAL p
1520 FOR p=y1 TO y2
1530 PRINTTAB(x,p)CHR$(c);CHR$157;
1540 NEXT p
1550 ENDPROC
1560 :
```

This procedure creates a column of control codes just like that for specifying foreground colours, but the column is now two characters wide because both the colour (parameter c) and the background code (157) have to be included on each line (and you will need to leave space for these).



The Demo program

If you intend to use coloured graphics on a coloured background then the codes for the background colour need to appear to the left of that for the foreground colour. If the codes for the background appear after those for foreground, only the background colour will show (see illustration of demo program). As an example, the following two lines will set a blue background and a yellow foreground for the whole screen:

```
PROCbackground(0,0,24,140)
PROCforeground(2,0,24,147)
```

The two background characters will be the first on each line (in positions 0 and 1) followed by the foreground code (in position 2).

PIXEL GRAPHICS

Up to now we have been dealing with a mode 7 screen in terms of rows of characters (25 lines of 40 characters). But each graphics character is made up of a two by three matrix of pixels, and it can be very useful to have a routine which places a single pixel at any point on the screen. Our co-ordinates will now be 80 pixels horizontally (numbered 0 to 79) and 75 pixels vertically (numbered 0 to 74), with (0,0) in the top left-hand corner as for text.

A procedure to do this is more complex than those we have seen previously, as it has to be written so that due recognition is made of any other pixels displayed adjacent to the one to be placed by the procedure. Thus the routine has to determine, from the pixel graphics co-ordinates, the character position on the screen, read and decode any existing graphics character in that position, incorporate the new pixel and write the character back to the screen. Here is the procedure.

```
 1800 DEF PROCpixel(x,y)
 1810 LOCAL char,x1,x2,y1,y2
 1820 x1=x DIV2:y1=y DIV3
 1830 x2=x MOD2:y2=y MOD3
 1840 char=FNchar(x1,y1):IF char>128 AND
char<160 THEN char=val(y2,x2) OR 160 EL
SE char=char OR val(y2,x2) OR 128
 1850 PRINTTAB(x1,y1)CHR$(char);
 1860 ENDPROC
 1870 :
 1900 DEF FNchar(x,y)
 1910 LOCAL A%,C
 1920 VDU31,x,y
 1930 A%=135:C=USR(&FFF4)
 1940 =(C AND &FFFF) DIV &100
```

The array val() holds the individual pixel values, and must be dimensioned and set up before PROCpixel is called in the main program (see lines 120 to 150 in the demo program listed at the end of this article.

Lines 1820 and 1830 calculate the correct character position. An additional function, FNchar(x,y), is called at line 1840 to find the existing character in this position. This can only be done with USR call shown, and I suggest you just accept this (it is covered in your User Guide in the section on OSBYTE calls). Depending on the character found, the new character is formed and then printed in line 1850. Thus:

```
PROCpixel(30,40)
```

would place a single pixel, in the current graphics foreground colour, in pixel position 30,40.

Once we have this procedure up and running we can use it to produce further effects. The next two procedures both use PROCpixel to draw either a vertical or a horizontal line. In each case the position and ends of the line are determined by the parameters.

Continued on page 56

A General Purpose Line-Input Function

Gareth Williams describes a highly versatile input routine which will smarten up your screen data input displays.

INTRODUCTION

It is often said that the things which make a good program are the routines that accept input from the user, and produce results on the screen or paper. This is understandable, because a good program should not only function well, but should look smart and above all appear robust. For example, if a prompt is displayed saying:

```
Please enter a number
```

and a naive user types 'Thirty six' in reply, then the program shouldn't simply crash with an error such as:

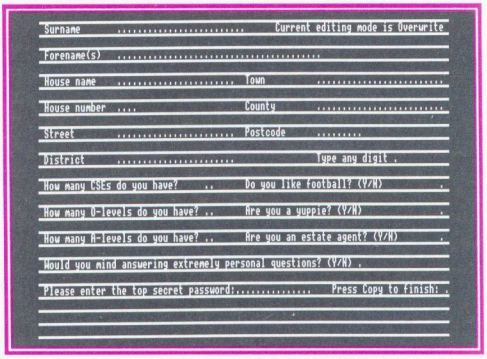
```
Type Mismatch at line 920
```

At the very least, the program should repeat the request, and it would be even more friendly if it could suggest a reason why the original input was rejected. Another possibility is only to accept characters that are valid for the chosen type of input. For example if a number is requested then the program would only respond to, and hence echo to the screen, the digits '0'-'9' and perhaps '-' and '.'.

Unfortunately, that is an ideal situation, and anybody who has written a substantial program will know that designing robust input routines is very time consuming, and nowhere near as satisfying as designing the main workings of the program. Therefore, many otherwise perfect programs resort to using Basic's INPUT statement with little or no checking on what the user enters. However, the machine code routine presented here helps to solve the problem by providing a robust data entry routine in which you can specify the type of input allowable. The routine deals with the job of showing the input on the screen in a neat way, and also editing the input line up to the point where Return is pressed. A facility also exists to allow brief help information on each entry item.

THE PROGRAM

Because of the length of the program, it is presented here in its entirety, together with a demonstration, but the full details on how the routine works, and how to use it in your own programs will not be covered until the next issue. To start with, enter and save listing 1. Running this program assembles the machine code which forms the heart of the input routine, and saves this with the filename 'EdLine'. Next, you should enter the demonstration program from listing 2, and save and run it. The first task this performs is to load in the 'EdLine' file, and therefore this must have been assembled with the first program beforehand.



The 'Demo' program in action

When you run the demonstration, you will be presented with a mode 3 screen containing a number of input prompts. Each of these is followed by a series of dots indicating how many characters will be accepted in the input. At the top right of the screen is a message indicating that you are in OverWrite mode. The cursor is positioned at the start of the first entry.

Movement within the line currently being edited is via the left and right cursor keys. When used by themselves these keys move the cursor left or right by one character. When used in conjunction with the Shift key, the cursor is moved to the start or the end of the input line.

When the Ctrl key is used with the left or right cursor keys, the line is deleted from the current cursor position to the start or end of the line respectively.

To move down to the next input line, press Return or Cursor-Down. To move up to the previous input line, press Cursor-Up. If you move off the top of the document, you will reappear at the bottom; if you fall off the bottom of the document, you will reappear at the top. Help messages may be obtained by pressing Ctrl-H, and these will appear for two and a half seconds at the bottom of the screen. To change to Insert mode press Ctrl-I (or Tab), and to revert back to OverWrite mode use Ctrl-O.

To finish entering details on this document, move to the final input on the screen ('Press Copy to finish:'), and press the Copy key. You will be returned to Basic. In a real situation the details entered would then be processed by the remainder of the program.

Next month we will conclude this article with details of how the input routine can be incorporated into your own programs. In the meantime, looking at the data statements at the end of the demonstration program should give you a general idea of how the input criteria are specified.

Listing 1

```
   10 REM Program EditLine
   20 REM Version B1.1
   30 REM Author  Gareth Williams
   40 REM BEEBUG  August/September 1989
   50 REM Program subject to copyright
   60 :
  100 MODE7
  110 PROCAssemble
  120 PROCSaveCode
  130 END
  140 :
 1000 DEF PROCSaveCode
 1010 F$="EdLine"
 1020 OSCLI("SAVE "+F$+" "+STR$~Put+" "+
STR$~O%+" "+STR$~Mc+" "+STR$~Mc)
 1030 ENDPROC
 1040 :
 1050 DEF FNStartOfCode
 1060 A%=133:X%=3
 1070 =((USR(OSByte) AND &FFFF00) DIV&10
0)-&500
 1080 DEF PROCAssemble
 1090 OSByte=&FFF4:OSRdCh=&FFE0
 1100 OSWrCh=&FFEE:OSAsci=&FFE3
 1110 OSArgs=&FFDA:Mc=FNStartOfCode
 1120 DIM Put &500
 1130 P%=&70:[OPT2
 1140 .Cursor   EQUB0
 1150 .Stat1    EQUB0
 1160 .Stat2    EQUB0
 1170 .Stat3    EQUB0
 1180 .Stat4    EQUB0
 1190 .XC       EQUB0
 1200 .YC       EQUB0
 1210 .MLen     EQUB0
 1220 .Len      EQUB0
 1230 .Flag1    EQUB0
 1240 .Flag2    EQUB0
 1250 .Flag3    EQUB0
 1260 .CurPos   EQUB0
 1270 .NumFlag  EQUB0
 1280 .Temp     EQUW0
 1290 .LineAddr EQUW0
 1300 .zp       EQUW0
 1310 .Dummy
 1320 ]
 1330 FOR Opt=4 TO 7 STEP 3
 1340 P%=Mc:O%=Put
 1350 [OPTOpt
 1360 STX zp:STY zp+1
 1370 LDY #(Dummy-Cursor-1)
 1380 .PushLoop
 1390 LDA Cursor,Y:PHA:DEY:BPL PushLoop
 1400 LDY#14:LDA(zp),Y:STA MLen:LDY#3
 1410 .FlagLoop
 1420 LDA(zp),Y:STA Len,Y
 1430 DEY:BNE FlagLoop:STY NumFlag
 1440 LDA#4:LDX#1:JSR OSByteWrite:STX Cu
rsor
 1450 LDA#225:LDX#1:JSR OSByteWrite:STX
Stat1
 1460 LDA#226:LDX#144:JSR OSByteWrite:ST
X Stat2
 1470 LDA#227:LDX#160:JSR OSByteWrite:ST
X Stat3
 1480 LDA#228:LDX#1:JSR OSByteWrite:STX
Stat4
 1490 LDA Flag3:AND#1:STA Flag3
 1500 LDY#10:LDA(zp),Y:STA LineAddr
 1510 INY:LDA(zp),Y:STA LineAddr+1
 1520 LDA Flag2:PHA:AND#2:BEQ NoClear
 1530 .SetZeroLen
 1540 LDY#0:LDA#13:STA(LineAddr),Y
```

```
1550 .NoClear
1560 PLA:AND#4:BEQ NoFlush
1570 LDA#15:LDX#&FF:JSR OSByte
1580 .NoFlush
1590 LDA Flag1:AND#6:BNE SingleKey
1600 LDA Flag1:AND#1:BEQ GetInitialLen
1610 .SingleKey LDY#1:STY MLen
1620 LDA#13:STA(LineAddr),Y
1630 LDA Flag1:AND#4:BEQ NotOnlySpecKey
1640 LDA Flag1:ORA#64:STA Flag1
1650 .NotOnlySpecKey
1660 LDA Flag2:ORA#1:STA Flag2
1670 LDA Flag1:ORA#1:STA Flag1
1680 LDA Flag1:AND#6:BEQ GetInitialLen
1690 LDA Flag2:AND#&E7:STA Flag2
1700 .GetInitialLen
1710 LDY#0
1720 .LenLoop
1730 LDA(LineAddr),Y:INY:BEQ SetZeroLen
1740 CMP#13:BNE LenLoop:DEY:STY Len
1750 CPY MLen:BCC LenOkay
1760 LDY MLen:STY Len
1770 LDA#13:STA(LineAddr),Y
1780 .LenOkay
1790 LDY#18:LDA(zp),Y
1800 CMP#&FF:BNE NotAtEnd:LDA Len
1810 .NotAtEnd
1820 STA CurPos
1830 LDY#4:LDA(zp),Y
1840 CMP#&FF:BEQ NoMovePrompt
1850 TAX:INY:LDA(zp),Y:TAY:JSR MoveXY
1860 .NoMovePrompt
1870 LDY#8:LDA(zp),Y:STA Temp
1880 INY:LDA(zp),Y:STA Temp+1
1890 ORA Temp:BEQ MoveInput
1900 JSR MessTemp
1910 .MoveInput
1920 LDY#6:LDA(zp),Y
1930 CMP#&FF:BEQ NoMoveInput
1940 TAX:INY:LDA(zp),Y:TAY
1950 JSR MoveXY
1960 .NoMoveInput
1970 LDA#134:JSR OSByte:STX XC:STY YC
1980 LDY#0:JSR DispLine
1990 LDA Flag2:BPL NotJustDisplay
2000 JMP Return
2010 .NotJustDisplay
2020 LDY CurPos:JSR MoveCur
2030 .ReadLoop
2040 JSR OSRdCh:BCC NotEscape
2050 LDA#126:JSR OSByte
2060 LDA Flag1:BMI ReadLoop
2070 LDY#0:STY Len
2080 LDA#13:STA(LineAddr),Y
2090 LDA Flag3:ORA#128:JMP Return
```

```
2100 .NotEscape
2110 LDX#0
2120 .FnLoop
2130 LDY FnTable,X:BEQ NotFn
2140 CMP FnTable,X:BEQ FnFound
2150 INX:INX:INX:BNE FnLoop
2160 .FnFound
2170 LDA FnTable+1,X:STA Temp
2180 LDA FnTable+2,X:STA Temp+1
2190 JMP(Temp)
2200 .NotFn
2210 PHA:LDA Flag1:AND#64:BEQ NoSpecK
2220 PLA:PHA
2230 LDY#15:CMP(zp),Y:BNE NoSpecK
2240 PLA:LDA Flag3:ORA#16:JMP Return
2250 .NoSpecK
2260 LDA Flag1:AND#32:BEQ NoHelpK
2270 PLA:PHA
2280 LDY#16:CMP(zp),Y:BNE NoHelpK
2290 LDY#12:LDA(zp),Y:STA Temp
2300 INY:LDA(zp),Y:STA Temp+1
2310 ORA Temp:BEQ NoHelpMess
2320 LDY#0:LDA(Temp),Y:TAX
2330 INY:LDA(Temp),Y:TAY:JSR MoveXY
2340 CLC:LDA Temp:ADC#2:STA Temp
2350 BCC NoHI:INC Temp+1
2360 .NoHI
2370 JSR MessTemp
2380 .NoHelpMess
2390 PLA:LDA Flag3:ORA#4:JMP Return
2400 .NoHelpK
2410 PLA:CMP#32:BCC NotValChar
2420 CMP#127:BCC ValidChar
2430 .NotValChar
2440 PHA:LDA Flag1:AND#16:BEQ NextChar
2450 PLA:LDY#17:STA(zp),Y
2460 LDA Flag3:ORA#8:JMP Return
2470 .ValidChar
2480 PHA:CMP#32:BNE NotSpace
2490 LDA Flag1:AND#8:BNE NextChar
2500 BEQ CharChecked
2510 .NotSpace
2520 LDA Flag1:AND#4:BEQ NotSpecOnly
2530 PLA:PHA
2540 LDY#15:CMP(zp),Y:BNE NextChar
2550 .NotSpecOnly
2560 PLA:JSR CheckChar:PHA:BCS NextChar
2570 .CharChecked
2580 LDA Flag3:AND#1:BEQ DoShuffle
2590 LDY CurPos:CPY Len:BNE NoShuffle
2600 CPY MLen:BEQ NextChar
2610 INC Len:INY:LDA#13:STA(LineAddr),Y
2620 BNE NoShuffle
2630 .DoShuffle:LDY Len
2640 CPY MLen:BEQ NextChar:INC Len
```

```
 2650 .Shuffle LDA(LineAddr),Y:INY:STA(L
ineAddr),Y:DEY:DEY:CPY#&FF:BEQ NoShuffle
:CPY CurPos:BCS Shuffle
 2660 .NoShuffle PLA:PHA:LDY CurPos:STA(
LineAddr),Y
 2670 LDY CurPos:JSR DispLine:INC CurPos
 2680 PLA:LDA Flag1:AND#1:BNE SKReturn
 2690 PHA
 2700 .NextChar PLA:JMP NotJustDisplay
 2710 .SKReturn LDA Flag3:ORA#32
 2720 .Return
 2730 LDY#3:STA(zp),Y
 2740 LDY#14:LDA Len:STA(zp),Y
 2750 LDY#18:LDA CurPos:STA(zp),Y
 2760 LDA#228:LDXStat4:JSR OSByteWrite
 2770 LDA#227:LDXStat3:JSR OSByteWrite
 2780 LDA#226:LDXStat2:JSR OSByteWrite
 2790 LDA#225:LDXStat1:JSR OSByteWrite
 2800 LDA#4:LDXCursor:JSR OSByteWrite
 2810 LDY#(Dummy-Cursor-1):LDX#0
 2820 .PullLoop
 2830 PLA:STA Cursor,X:INX:DEY
 2840 BPL PullLoop
 2850 RTS
 2860 \--------------------------------
 2870 .OSByteWrite
 2880 LDY#0:JMP OSByte
 2890 .MoveXY
 2900 JSR CurOff:LDA#31:JSR OSWrCh
 2910 TXA:JSR OSWrCh
 2920 TYA:JSR OSWrCh:JMP CurOn
 2930 .MessTemp
 2940 JSR CurOff:LDY#0
 2950 .MTLoop
 2960 LDA(Temp),Y:INY:BEQ MTRet
 2970 CMP#13:BEQ MTRet
 2980 JSR OSWrCh:JMP MTLoop
 2990 .MTRet JMP CurOff
 3000 .DispLine
 3010 JSR MoveCur:JSR CurOff
 3020 .DispLoop:LDA(LineAddr),Y
 3030 CMP#13:BEQ BlankChar
 3040 JSR Hide:JSR OSWrCh
 3050 INY:BNE DispLoop
 3060 .BlankChar
 3070 TYA:PHA:LDX#32
 3080 LDA Flag2:AND#64:BEQ NoDot:LDX#46
 3090 .NoDot
 3100 CPY MLen:BEQ EndDisp
 3110 TXA:JSR OSWrCh:INY:BNE NoDot
 3120 .EndDisp
 3130 JSR CurOn:PLA:TAY:RTS
 3140 .Hide
 3150 PHA:LDA Flag2:AND#32:BEQ NoHide
 3160 PLA:LDA#42:PHA
 3170 .NoHide PLA:RTS
 3180 .MoveCur
 3190 TYA:PHA:CLC:ADCXC:TAX:LDY YC
 3200 JSR MoveXY:PLA:TAY:RTS
 3210 .FnTable
 3220 OPT FNEntry(136,Left)
 3230 OPT FNEntry(137,Right)
 3240 OPT FNEntry(138,Down)
 3250 OPT FNEntry( 13,Down)
 3260 OPT FNEntry(139,Up)
 3270 OPT FNEntry(127,Delete)
 3280 OPT FNEntry(  9,Insert)
 3290 OPT FNEntry( 15,OverWrite)
 3300 BRK
 3310 .ShiftControl
 3320 LDX#&FF:JSR KeyboardScan:BEQ Shift
Down
 3330 LDX#&FE:JSR KeyboardScan:BEQ Contr
olDown
 3340 LDA#0:RTS
 3350 .ShiftDown
 3360 LDA#&FF:RTS
 3370 .ControlDown
 3380 LDA#1:RTS
 3390 .KeyboardScan
 3400 LDY#&FF:LDA#129:JSR OSByte:CPX#&FF
:RTS
 3410 .Left
 3420 JSR ShiftControl
 3430 BEQ JustLeft:BMI SLeft:BPL CLeft
 3440 .JustLeft
 3450 LDA CurPos:BEQ ReturnFn
 3460 DEC CurPos:LDA#8:JSR OSWrCh
 3470 .ReturnFn
 3480 JMP ReadLoop
 3490 .SLeft
 3500 LDY#0:STY CurPos
 3510 .ShiftReturn
 3520 JSR MoveCur:JMP ReadLoop
 3530 .CLeft
 3540 LDY CurPos:BEQ ReturnFn
 3550 .ShiftLeft
 3560 LDA(LineAddr),Y:PHA
 3570 TYA:SEC:SBCCurPos:TAY
 3580 PLA:STA(LineAddr),Y
 3590 CMP#13:BEQ EndShiftLeft
 3600 TYA:CLC:ADC CurPos:TAY:INY:BNE Shi
ftLeft
 3610 .EndShiftLeft
 3620 LDA Len:SEC:SBCCurPos:STA Len
 3630 LDY#0:STY CurPos
 3640 .CtrlReturn
 3650 JSR DispLine:LDY CurPos:JMP ShiftR
eturn
 3660 .Right
```

```
 3670 JSR ShiftControl
 3680 BEQ JustRight:BMI SRight:BPL CRigh
t
 3690 .JustRight
 3700 LDA CurPos:CMP Len:BEQ ReturnFn
 3710 INC CurPos:LDA#9:JSR OSWrCh
 3720 JMP ReadLoop
 3730 .SRight
 3740 LDY Len:STY CurPos
 3750 JMP ShiftReturn
 3760 .CRight
 3770 LDA#13:LDY CurPos:STA(LineAddr),Y
 3780 STY Len
 3790 JMP CtrlReturn
 3800 .Down
 3810 LDA Flag3:ORA#32:JMP Return
 3820 .Up
 3830 LDA Flag3:ORA#64:JMP Return
 3840 .Insert LDA Flag3:BEQ ReturnFn
 3850 LDA#2:STA Flag3:JMP Return
 3860 .OverWrite LDA Flag3:BNE OWRet
 3870 LDA#3:STA Flag3
 3880 .OWRet
 3890 JMP Return
 3900 .Delete
 3910 LDY CurPos:BNE IsDP:JMP ReturnFn
 3920 .IsDP DEY:LDA(LineAddr),Y:INY
 3930 CMP#ASC".":BNE ShuffleD
 3940 LDA NumFlag:AND#&7F:STA NumFlag
 3950 .ShuffleD LDA(LineAddr),Y:DEY:STA(
LineAddr),Y
 3960 INY:INY:CMP#13:BNE ShuffleD
 3970 DEC Len:DEC CurPos:LDY CurPos:JMP
CtrlReturn
 3980 .CheckChar
 3990 PHA
 4000 LDA Flag2:AND#24:BNE Numeric
 4010 LDA Flag2:AND#1:BEQ NoCapify
 4020 PLA:CMP#ASC"a":BCC NoCap:CMP#ASC"z
"+1:BCS NoCap
 4030 AND#&DF
 4040 .NoCap PHA
 4050 .NoCapify LDA Flag1:AND#2:BEQ RetC
CP
 4060 PLA:CMP#ASC"Y":BEQ RetCC:CMP#ASC"N
":BEQ RetCC
 4070 .RetCS SEC:RTS
 4080 .RetCCP PLA
 4090 .RetCC CLC:RTS
 4100 .Numeric PLA
 4110 CMP#ASC"0":BCC NotDigit
 4120 CMP#ASC"9"+1:BCC RetCC
 4130 .NotDigit PHA:LDA Flag2
 4140 AND#16:BNE ExtraNumeric:PLA:SEC
 4150 :RTS
 4160 .ExtraNumeric
 4170 PLA:CMP#ASC"-":BEQ FirstPos
 4180 CMP#ASC"+":BEQ FirstPos
 4190 CMP#ASC".":BNE RetCS
 4200 LDXNumFlag:BMI RetCS
 4210 PHA:TXA:ORA#128:STA NumFlag:PLA
 4220 BNE RetCC
 4230 .FirstPos
 4240 LDY CurPos:BNE RetCS:BEQ RetCC
 4250 .CurOn LDA#1:BNE CurOnOff
 4260 .CurOff LDA#0
 4270 .CurOnOff PHA:LDA#23:JSR OSWrCh
 4280 LDA#1:JSR OSWrCh:PLA:JSR OSWrCh
 4290 TXA:PHA:LDX#7:LDA#0
 4300 .Zeroes JSR OSWrCh:DEX:BNE Zeroes
 4310 PLA:TAX:RTS
 4320 ]:NEXT
 4330 ENDPROC
 4340 DEF FNEntry(V%,A%):LOCALo%
 4350 o%=2*(Opt DIV2)
 4360 [OPTo%:EQUBV%:EQUWA%:]
 4370 =Opt
```

Listing 2

```
   10 REM Program Input Function Demo
   20 REM Version B1.00
   30 REM Author  Gareth Williams
   40 REM BEEBUG  August/September 1989
   50 REM Program subject to copyright
   60 :
  100 MODE3:HIMEM=HIMEM-&500
  110 PROCLoad
  120 PROCDemo
  130 END
  140 :
 1000 DEF PROCLoad
 1010 F$="EdLine"
 1020 OSCLI("LOAD "+F$)
 1030 Mc=HIMEM
 1040 ENDPROC
 1050 :
 1060 DEF FNEditLine(P$,PX%,PY%,I$,IX%,I
Y%,L%,H$,F1%,F2%,Disp%)
 1070 ParBlk?1=F1%
 1080 ParBlk?2=F2%-128*Disp%
 1090 IFDisp% ParBlk?3=0 ELSEParBlk?3=Pa
rBlk?3 AND1
 1100 ParBlk?4=PX%
 1110 ParBlk?5=PY%
 1120 ParBlk?6=IX%
 1130 ParBlk?7=IY%
 1140 ParBlk!8=PromptMessage
 1150 ParBlk!10=InputBuffer
```

Continued on page 50

The Comms Spot

by Peter Rochford

This month the Comms Spot is a mixture of news items and general information on what's happening on the comms scene at the moment.

The Micronet database on Prestel has recently undergone a change of ownership. Formerly it was controlled by Telemap and was part-owned by British Telecom. Now it has become wholly owned by BT and forms part of their Dialcom subsidiary. To coincide with this change, Micronet has moved its headquarters from the smoke and grime of the city at Herbal Hill in London, to the new Dialcom offices near rural Apsley in Hertfordshire. Their full address is now: Dialcom House, Brindley Way, Hemel Hempstead, Herts HP3 9RR, and their telephone number is (0442) 237788.

Still on the subject of Micronet, they now have a very active BBC Microbase area on their database thanks to the efforts of the new editor, Paul Vigay. Do give it a look next time you log-on as there is plenty of interest for most Beeb and Arc owners. I must comment that it does have a fair bias towards the Arc but then this is only natural. The new telesoftware gateway problems now seem to have been resolved and there are plenty of free programs to download for Beeb owners, with a growing quantity for the Arc, too.

Prestel has recently announced details of forthcoming changes to its Mailbox Email facility. So extensive are the changes and extra facilities, that subscribers have been sent a small booklet outlining their use. Space here does not permit a full run down on what is to be offered, but this includes multiple mailboxing using a mailing list of recipients, notification of receipt, multi-frame mailboxes and the ability to automatically download all your mailboxes to one file when using a micro with the right kind of software. Of course good old Prestel is not providing all these new goodies free of charge. However, use of most of the existing mailbox facilities will remain free. Perhaps when the new system is under way, I will take a closer look at the developments in a future Comms Spot.

Sadly, I have to report that the Viewtel database on Prestel has just announced that it is not intending to renew its contract with Prestel. Instead, it is to transfer its operation to the Istel network. Viewtel state that its withdrawal is due to changes in Prestel's policy towards the operation of Information Providers (IPs), with which it cannot, or is not, prepared to comply. Viewtel have been on Prestel since its inception and can justifiably claim to have been the first company in the UK to provide 'an electronic newspaper'. Their pages offered a mixture of news of all kinds along with general features and entertainment.

This move by Viewtel follows other notable departures from Prestel over the last year or two, due to the changes in Prestel's policies towards IPs. Amongst these was the Viewfax database which catered largely for BBC micro users. Remember Ramlink and Tubelink that provided such a great service to Beeb owners ?

Another major IP that chose to quit for the same reasons was Timefame, who were also an early pioneer in the field of viewdata. Their lively and popular database provided many areas of interest and in particular, did much for the users of micros. Timefame went on to set up its own online system called Epnitex which I have mentioned before in the Comms Spot. I am sure that Viewtel will be very much missed as was Timefame.

Personally, I am disappointed and surprised that Prestel continues to make decisions that result in companies, who have provided such an interesting and important contribution to Prestel and formed part of its success, to leave. Who's next ?

Surely the departure of these databases only weakens and dilutes what Prestel offers the majority of domestic users, and will lead to a decline in the number of home users. Already, the decision by Prestel to increase its charges, and the increase in Micronet charges, have led to a significant number of home micro users abandoning the service altogether. Unless Prestel rethinks its policy, this trend will, I am sure, continue and the future begins to look bleaker all the time.

On the subject of departures, it is not only Prestel that suffers from them. Dialcom's other online system, Telecom Gold, used to host the Microlink database belonging to Database publications which catered for micro enthusiasts whilst providing low cost access to Gold's Email facility and other services. Microlink has now decided to change like Viewtel, to the Istel system, again due to Dialcom's policies and attitudes. With the migration of so many of the better services to Istel, I will shortly take a look at Istel in the Comms Spot and report on what else it has to offer.

Those who use Telecom Gold will be interested to know that at long last moves are being made to provide access at 2400 baud. At the moment, users in London with the luxury of V22bis modems can sample high-speed Gold on a test number; 01-203 3033. When the service is fully operational, it will offer MNP error correction to level 5. Along with this, you will be able also to download and upload using a number of error-checked protocols including Xmodem.

I recently did a feature on SID, Acorn's own information database. The news is that SID continues to expand and is now operating with new custom written software running on Archimedes computers. It is, I believe, the intention in the future to provide a system of networked Arcs for SID, where each user who dials in will be connected to his own individual Arc! Further rumour has it that in the not too distant future, access to SID will be by gateway from Micronet's database on Prestel. At the moment, unless you are a subscriber to SID and use the Fastrak network, you have to dial direct to Cambridge for access. If the gateway story is true, then we can hopefully look forward to cheap local access and a really useful facility for users of Acorn machines. The only question is, whether there is going to be some heavy charges for accessing the service. Let's hope not.

See you next month!

Daisy-Chained ROMs

Sebastian Lazareno and David Spencer show how you can make more efficient use of sideways RAM.

With the increased ownership of sideways RAM, helped a lot by its inclusion in the Master and Compact, there has been a corresponding increase in programs that run as ROM images, both in BEEBUG and other magazines. However, most of these programs are at most 2K long, and it seems very extravagant to tie up an entire 16K bank of sideways RAM with each one. The ideal solution would be to collect together a number of such ROM images, and arrange them so that they can all be loaded into the same sideways RAM bank. Fortunately, this can be done relatively easily, and the technique will be described here together with the pitfalls to look out for.

Before explaining the technique, we need to take a quick look at the format of a sideways ROM, or more particularly, the header that is placed at the start of a ROM. Full details of this can be found in the Master reference manual part 1, or the Advanced User Guide. However, all we need to know is that the first six bytes of the ROM image (at addresses &8000 to &8005) form two jump instructions. The first of these is the *language entry point*, and is used to enter the ROM if it is being started up as the current language. The second jump (at &8003) is the more important, and forms the *service entry point*. The operating system calls this for tasks such as claiming workspace, and interpreting unrecognised star commands.

The way in which we will combine several ROM images into one is to make the service entry of each ROM call the service entry of the next one before performing its own processing. In this way, the service entry of each constituent ROM will be called in turn, with the last ROM in the list executing its service code first. To see how this can be done, we will take as an example the Versatile ROM Manager from BEEBUG Vol.7 No.8 (itself a ROM image), and link this to Disc Access ROM from the Workshop article in Vol.8 No.2. Listing 1 shows the relevant section from the source code of the ROM Manager.

The three BRK instructions in line 160 signify that the ROM has no language entry point, while the

```
130 FOR opt=4 TO 7 STEP 3
140 P%=&8000:O%=HIMEM:Q%=O%
150 [ OPT opt
160 BRK:BRK:BRK
170 JMP service
    .
    .
280 .service
290 CMP #1:BEQ service1
300 CMP #4:BEQ service4:RTS
```

Listing 1. Fragment of ROM Manager

JMP in line 170 is the service entry jump. This jumps to the routine starting at line 280 which decodes the various service calls. All we need to do is assemble the second ROM image (the Disc Access ROM in this case) to start immediately after the first ROM (the ROM manager). We will then link the service handler of the first image into the second. This is shown in figure 1. To link the two images, we will add the instruction:

```
JSR image2+3
```

as the first instruction of the service routine in the first ROM image (the ROM manager). The value *image2* is the address at which the second ROM image is assembled (as shown in figure 1).

Now, when a service call is issued, our combined ROM image will be entered at location &8003 (assuming no higher priority ROM claims the call). This then jumps to the service routine proper, which in turn immediately calls the service entry point of the second image. The second ROM image will perform all its service handling, and execute an RTS when in has finished. In a normal ROM this would return to the operating system. However, in our linked images it returns to the service routine of the first ROM image, which subsequently executes and then returns to the operating system.

PUTTING INTO PRACTICE

The best way to clarify this daisy-chaining process is actually to try it out with our chosen ROM images. Incidentally, the original programs for both the ROM manager and the Disc Access ROM are provided on this month's magazine disc with which to experiment.

The first step is to find out the length of the ROM manager image. To do this, load the source code and run it. This will assemble the ROM image and leave the code pointer P% pointing to the first location after the assembler program. In this case, printing the value of P% (in hex) should give the result &8279. However, we need to add the JSR instruction to call the second ROM image, so this will increase the length by three bytes. Therefore, the second ROM image will need to be assembled to start at location &827C, and the service code of the first image must perform a JSR &827F instruction. We can now add the line:

```
285 JSR &827F
```

to the ROM manager source code and reassemble it. This will automatically save the ROM image with the name 'MANAGER'. As a double check, you can print the value of P% and make sure that it is indeed &827C.

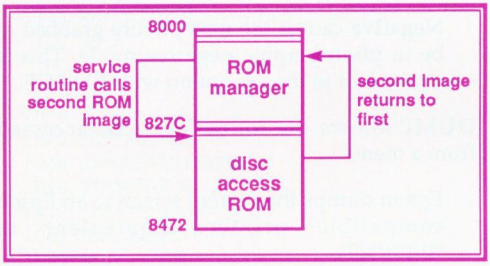


Figure 1. Chaining two ROM images together

Having done this, the second ROM image now needs to be assembled. To do this, load the source code for the Disc Access ROM and change line 150 to read:

```
150 P%=&827C:O%=code
```

and run the program. This will assemble the ROM image to start at address &827C (rather than &8000), and save it to disc as 'DiscROM'.

The final stage in the process is to physically link together the two ROM images. This is best done by loading the images into memory one after another, and then saving them as a single image. To do this you need to find the lengths of the two saved images (using *INFO), and then load them into memory and save the result. For example, with our two images, this could be done using:

```
*LOAD Manager 1000
*LOAD DiscROM 127C
*SAVE Combined 1000 +472
```

The combined ROM image can then be loaded in the same way as any other ROM.

So far we have only considered two ROM images, buts clearly the technique can be extended to as many images as will fit in the 16K of a single ROM. In this case, the first image calls the second, which immediately calls the third and so on.

POSSIBLE PROBLEMS

There are a number of limitations with this techniques which must always be considered. First of all, only the first ROM in the daisy-chain can have a language entry. This will not in general cause many problems because the vast majority of 'short' ROMs will be service-only types.

The second consideration is the priority of ROMs. Normally, ROMs in higher numbered positions have higher priority which means that they receive unrecognised star commands and the like before the lower ROMs. In a combined ROM image, the first constituent image has the lowest priority, and the last has the highest. This is because of the way that service calls 'leap' through to the last image, and then 'ripple' back through each of the service routines.

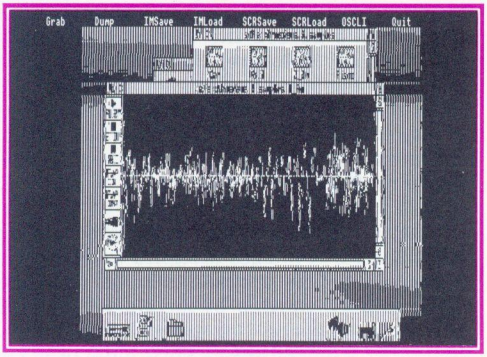
Another possible problem arises from the fact that many ROM images designed to be used in sideways RAM (as opposed to ROM) make use of the unused portion of the sideways RAM area for workspace. For example, the Partial Renumber from BEEBUG Vol.7 No.7 uses the workspace immediately after the ROM image in which to store the program's line numbers. If such a ROM is included in a daisy-chain then it will need to be modified so that it uses workspace positioned after all the other images, rather than immediately after itself.

A final problem to look out for is a ROM which claims private workspace using service call &22 (or 2 on a model B). ROMs claiming memory in this way store a pointer to their workspace in location &DFx, where x is the ROM number in hex. Clearly, when several ROM images exist in a single ROM slot, they can't all use this one location. In this case it will be necessary to alter the individual images so that one claims as much workspace as is needed by all of them, and then each uses its own portion of this.

Using a Video Digitiser

Rupert Thompson offers a program to simplify using the Watford Electronics Video Digitiser.

The Watford Electronics Video Digitiser is a very useful upgrade for the humble model B. However, the ROM based commands provided to operate the system are intended to be used in bigger programs. One reason for this is that if you grab an image in immediate mode, it will be ruined by the prompt being printed before you can save the screen. The program presented here provides a front-end for the digitiser, as well as simplifying the setup procedure. Unfortunately, the program only works in MODE 0, and the top display line is lost for the menu, although this is allowed for when dumping and saving the picture.



The Digitiser utility in use

Start off by entering the listing and saving it. Before running it, make sure that the digitiser ROM is installed and active, and that the digitiser is connected to both the computer and a suitable camera. The red 'Sync' light on the digitiser should flash periodically to indicate this. If it doesn't, check that everything is connected and powered up, and make sure that you are using the video out and not RF out sockets on the camera. When the program is RUN you see eight options which are selected by moving the cursor to that which you require and pressing the Space Bar. In subsequent menus, ESCAPE returns you here, although this will not work at other times.

GRAB gives a further menu. From here the following options are available:

Snapshot grabs a single picture at the next sync pulse. (Equivalent to *IMAGE).

Multiple continuously grabs images until the Space Bar is pressed.

Delay makes the computer wait a specified length of time before an image is grabbed. This lets the operator take a picture of himself, and is equivalent to the option available on some still cameras. A delay of about fifteen seconds is usually adequate.

Negative causes the next picture grabbed to be in photographic negative mode. This is equivalent to the 'N' option with *IMAGE.

DUMP allows the following to be accessed from a menu:

Epson dumps the current screen to an Epson compatible printer (Equivalent to *IMPRNT).

Sparkjet dumps to an Acorn/Olivetti Sparkjet printer (Equivalent to *OPRINT).

Aspect allows the Aspect Ratio of the dump to be altered, and defaults to 100. (See page 9 of the Digitiser Operating Manual for more information).

IMSAVE and **IMLOAD** are the equivalents of *IMSAVE and *IMLOAD, saving and loading screens in compressed form. However, such screens are of no use to packages such as AMX Pagemaker, so the options **SCRSAVE** and **SCRLOAD** save and load ordinary, uncompressed MODE 0 screens.

A final word about using digitised images with art programs - they will expect the image to be saved as a normal screen (use the SCRSAVE option. For an existing screen, the program can load it using IMLOAD, and resave it with

SCRSAVE). In addition AMX Pagemaker (Stop Press) requires that the image be a photographic negative. Thus you should use the Negative option (*IMAGE N) when grabbing a picture for Pagemaker, or if you wish to use a screen which was previously grabbed not as a negative, you can load the screen into Pagemaker, open a window around it and use the Invert Window option to convert it.

```
  10 REM Program Digitiser Front End
  20 REM Version B1.0
  30 REM Author  Rupert Thompson
  40 REM BEEBUG  August/September 1989
  50 REM Program subject to copyright
  60 :
 100 MODE0
 110 PROCvar
 120 ON ERROR CLS:REPORT:END
 130 exit%=FALSE
 140 REPEAT:PROCmenu:UNTIL exit%
 150 MODE7
 160 *FX229
 170 END
1010 :
1020 DEFPROCvar
1030 c$=CHR$13+CHR$127
1040 DIMmenu$(8)
1050 im$="":asp$="100"
1060 VDU23,1,0;0;0;0;
1070 *FX229,255
1080 ENDPROC
1090 :
1100 DEFFNbar(x,y,n,m$,f,l):LOCALI,t,c$
1110 *fx4,1
1120 PROCextract(m$,n)
1130 IF f THEN PROClist
1140 Q=0:nv=0:REPEAT:COLOUR129:COLOUR0:
PRINTTAB(x,y+Q);" ";menu$(Q);SPC(l-LENme
nu$(Q));
1150 REPEAT:G=GET
1160 IFG=&8B THENnv=Q-1
1170 IFG=&8A THENnv=Q+1
1180 IFnv<0 THENnv=0 ELSEIFnv>n THENnv=
n
1190 UNTILG=&8A ORG=&8B ORG=32 ORG=&88
1200 COLOUR128:COLOUR1:PRINTTAB(x,y+Q);
" ";menu$(Q);SPC(l-LENmenu$(Q)):Q=nv:UNT
ILG=32 ORG=&88:IFG=&88 THENQ=100
1210 *fx4
1220 =Q
1230 :
1240 DEFPROClist:COLOUR1:COLOUR128
1250 LOCALI
1260 FORI=0 TOn:PRINTTAB(x+1,y+I);menu$
(I):NEXTI:ENDPROC
1270 :
1280 DEFFNgetcha(v$):REPEAT:G=GET:UNTIL
INSTR(v$,CHR$G)>0 ORv$="" ORINSTR(c$,CHR
$G)>0:=G
1290 :
1300 DEFFNenter(x,y,l,o$,v$,p$)
1310 c$=CHR$13+CHR$&88+CHR$&8A+CHR$&8B+
CHR$9+CHR$&87+CHR$127
1320 LOCALi$:i$=o$
1330 *fx4,1
1340 COLOUR0:COLOUR129:PRINTTAB(x,y);"
";p$;SPC(l+2)
1350 x=x+LENp$+2:PRINTTAB(x,y);o$;
1360 REPEAT:G=FNgetcha(v$)
1370 IFG=127 ANDLENo$>0 THENo$=LEFT$(o$
,LENo$-1):PRINTTAB(x+LENo$+1,y);CHR$127;
1380 IFG>31 ANDG<127 ANDLENo$<l THENo$=
o$+CHR$G:PRINTCHR$G;
1390 IFG=&87 THENo$="":PRINTTAB(x,y);SP
C(l);TAB(x,y);
1400 IFG=9 ANDo$="" THENo$=i$:PRINTTAB(
x,y);o$;
1410 UNTILG=13 OR(G>&87 ANDG<&8C)
1420 flag=1:IFG=13 ORG=&8A THENflag=-1
ELSEIFG=&89 THENflag=-100
1430 *fx4
1440 COLOUR1:COLOUR128:PRINTTAB(x-LENp$
-2,y);" ";p$;" ";o$;SPC(l+1-LENo$)
1450 =o$
1460 :
1470 DEFPROCextract(m$,n)
1480 LOCALt,I:t=0:FORI=0TOn:menu$(I)=""
:REPEAT:t=t+1:c$=MID$(m$,t,1):menu$(I)=m
enu$(I)+c$:UNTILc$="*":menu$(I)=LEFT$(me
nu$(I),LENmenu$(I)-1):NEXTI:ENDPROC
1490 :
1500 DEFFNlotus(m$,n)
1510 LOCALQ,I
1520 PRINTTAB(0,0);SPC(80);
1530 *fx4,1
1540 PROCextract(m$,n):FORI=0 TOn:PRINT
TAB(I*10,0);menu$(I):NEXTI
1550 Q=0
1560 REPEAT:REPEAT:COLOUR0:COLOUR129:PR
INTTAB(Q*10,0);menu$(Q);SPC(9-LENmenu$(Q
));:G=GET:UNTIL G=&88 ORG=&89 ORG=32 ORG
=27
1570 COLOUR1:COLOUR128:PRINTTAB(Q*10,0)
;menu$(Q);SPC(9-LENmenu$(Q));
1580 IFG=&88 ANDQ>0 THENQ=Q-1 ELSE IF G
=&89 ANDQ<n THENQ=Q+1
1590 UNTILG=32 ORG=27
1600 *fx4
```

```
 1610 IFG=32 THEN=Q ELSE=-1
 1620 :
 1630 DEFPROCmenu:LOCALQ
 1640 Q=FNlotus("Grab*Dump*IMSave*IMLoad
*SCRSave*SCRLoad*OSCLI*Quit*",7)
 1650 IFQ=0 THENPROCgrab
 1660 IFQ=1 THENPROCdump
 1670 IFQ=2 THENPROCimsave
 1680 IFQ=3 THENPROCimload
 1690 IFQ=4 THENPROCscrsave
 1700 IFQ=5 THENPROCscrload
 1710 IFQ=6 THENPROCstar
 1720 IFQ=7 THENexit%=TRUE
 1730 ENDPROC
 1740 :
 1750 DEFPROCgrab:LOCALQ
 1760 im$=""
 1770 REPEAT
 1780 Q=FNlotus("Snapshot*Multiple*Delay
*Negative*",3)
 1790 IFQ=0 THENOSCLI("WIMAGE "+im$)
 1800 IFQ=1 THENREPEAT:OSCLI("WIMAGE "+i
m$):UNTIL INKEY(30)=32
 1810 IFQ=2 THENPROCdelay
 1820 IFQ=3 THENPROCnegative
 1830 UNTILQ<3
 1840 ENDPROC
 1850 :
 1860 DEFPROCnegative
 1870 IFim$="N" THENim$="" ELSE im$="N"
 1880 ENDPROC
 1890 :
 1900 DEFPROCdelay:PRINTTAB(0,0);SPC(79)
;
 1910 D%=VALFNenter(0,0,2,"","0123456789
","Delay (Secs):")
 1920 PRINTTAB(0,0);SPC(80);
 1930 PRINTTAB(0,0);"Counter: ";:TIME=0:
REPEAT:PRINTTAB(10,0);SPC(2);TAB(10,0);(
(D%*100)-TIME)DIV100:UNTILTIME=D%*100
 1940 OSCLI("WIMAGE "+im$):ENDPROC
 1950 :
 1960 DEFPROCdump:LOCALQ
 1970 REPEAT
 1980 Q=FNlotus("Epson*Sparkjet*Aspect*"
,2)
 1990 IFQ=0 THENPROCprint("WIMPRNT")
 2000 IFQ=1 THENPROCprint("WOPRINT")
 2010 IFQ=2 THENasp$=FNasp
 2020 UNTILQ<2
 2030 ENDPROC
 2040 :
 2050 DEFFNasp
 2060 LOCALv$
 2070 PRINTTAB(0,0);SPC(80);
 2080 REPEAT:v$=FNenter(0,0,3,asp$,"0123
456789","Aspect Ratio:"):UNTILVALv$>0 AN
DVALv$<=200
 2090 =v$
 2100 :
 2110 DEFPROCprint(p$)
 2120 PRINTTAB(0,0);SPC(80):PRINTTAB(0,0
);"Press any key to begin dumping ...";:
G=GET:COLOUR129:PRINTTAB(0,0);SPC(80);:C
OLOUR128
 2130 OSCLI(p$+" "+asp$)
 2140 ENDPROC
 2150 :
 2160 DEFPROCimsave
 2170 f$=FNfile
 2180 PRINTTAB(0,0);SPC(80);
 2190 IFf$<>"" THENOSCLI("WIMSAVE "+f$)
 2200 ENDPROC
 2210 :
 2220 DEFPROCimload
 2230 f$=FNfile
 2240 IFf$<>"" THENOSCLI("WIMLOAD "+f$)
 2250 ENDPROC
 2260 :
 2270 DEFFNfile:LOCALf$
 2280 PRINTTAB(0,0);SPC(80);
 2290 f$=FNenter(0,0,9,"","","File-name:
")
 2300 PRINTTAB(0,0);SPC(80);
 2310 =f$
 2320 :
 2330 DEFPROCscrsave
 2340 f$=FNfile
 2350 IFf$<>"" THENOSCLI("SAVE "+f$+" 30
00 7FFF")
 2360 ENDPROC
 2370 :
 2380 DEFPROCscrload
 2390 f$=FNfile
 2400 IFf$<>"" THENOSCLI("LOAD "+f$+" 30
00")
 2410 ENDPROC
 2420 :
 2430 DEFPROCstar
 2440 CLS:REPEAT
 2450 PRINT':INPUT"*"m$:PRINT:OSCLI(m$)
 2460 PRINT'"Press any key to continue";
 2470 UNTILGET$<>"*"
 2480 CLS:ENDPROC
 2490 :
 2500 DEFPROCerror
 2510 PRINTTAB(0,0);SPC(80):REPORT
 2520 PRINT". Press any key";:G=GET
 2530 ENDPROC
```

512 Forum

by Robin Burton

Since writing the last Forum I've been busy copying BBC BASIC discs and several points arise which are worth mentioning, so there's a slight change of plan this month.

RULES OF PLAY

First I must thank everyone who included a letter with their disc. Even if it was only a note to observe the formalities, the courtesy was much appreciated. It helped to make the job much more pleasant and personal.

The second point is that a number of Forum readers are recent converts to the 512 and/or to BEEBUG. As many said, it's the only magazine to cater for the 512 regularly. And so shall it continue with the enthusiasm you've shown. You might like to know that almost 140 copies of BBCBASIC have been dispatched to date, and more discs are still arriving.

Newer readers might therefore have missed these items which were included about six months ago, so here they are again. Bear with me because I may have another offer in a month or two.

You may not all realise it but I do the disc copying and letter answering myself. I'm not employed by BEEBUG and it makes a difference. If you write to me and expect a direct reply these are the rules. You must supply adequate return postage and packaging. Packaging should either be self addressed, or should include a suitable label, with sufficient postage to cover any items to be returned.

Quite a few will have had excess postage to pay. Sorry, but if I offer free software and a free copying service, I do not pay the postage as well.

Overseas readers can send international postage coupons instead of stamps - get them from your post office. As you don't know our postal rates just send the same value as it cost you to write to me. If it's not exactly right I'm not too worried. As I said last time, all I ask is that you make the effort.

For example Heinrich Lamm generously sent 10 Deutschmarks saying have a drink with the change. I did Heinrich, especially welcome as it was so warm, but the bank also charged me half its value to change it. Adrian van der Veen sent an extra disc instead of postage, which I thought was a reasonable offer. Professor Isaacson sent a sterling bank draft from Capetown, also all right, except it was payable to BEEBUG not to me, so a minor complication there.

In fact the majority of you did things right, for which I thank you. With this number of discs to copy and return anything that simplifies the job is welcome. A minute or two saved on each one, even writing out your address, soon adds up to a considerable amount of time. This leads me nicely to the next point.

DISC FORMATS

This time I'm not complaining, especially since I didn't specify a particular 512 disc format in the article, but after copying a few discs lately I found some of the figures interesting. Perhaps you might think so too - I'd never timed such an operation before.

So as to make the copying operation as fast as possible I created a 300K RAM disc and put all the BBCBASIC files into it, which left 68K of RAM disc unused. Point number one - although the files actually contain 186K or so of real data, the files occupy 232K plus. The minimum DOS file allocation size (one cluster) accounts for the 'missing' 46K.

This is worth remembering if you want to find the amount of space occupied by a directory or a group of files on disc. The normal 'DIR' command gives you only the total free space remaining for the whole disc, not the amount used (or left) by a directory.

'SDIR' will give you the space used by the contents of a directory, but this time it's the total size of the data in the files, not the total disc space allocated to them. As in the BBCBASIC copying exercise, it's easy to 'lose' 50K or so. The more small files there are, the bigger the total discrepancy.

It may seem a small point, but I created a batch file to set up the disc copy, so I allowed 300K for the RAM disc simply because I knew it would be more than big enough. There's just no straightforward way to find out exactly how much space is needed, other than adding together the sizes of all 59 files, rounding up each as appropriate, so I didn't.

DISC SPEED

A RAM disc is by far the best way to do this sort of mass copying job, because it's several times quicker than a winchester (no tube, you see) and many, many times faster than floppies. The first figure I noted was the time to copy the files to the RAM disc from my 800K master disc. This was the first job for each copy session and it took 48 seconds.

Most of you sent an 800K disc - not very surprising, we all know it's the quickest format, don't we? Well, it seems we don't. I had several 360K discs, two of 720K and perhaps a dozen or more of 640K. Thankfully the rest were 800K.

As I said, I've never bothered to time the different formats before, I know that 800K is the quickest and use nothing else unless there are special reasons. Well, when you've got 15 or 20 discs a day to copy you have to find something else to do, so I timed each of the different formats, especially after coming across the first 640K disc (to be honest at first I thought it was faulty). Here are the results:

800K - 2 mins 12 secs (1)
720K - 3 mins 51 secs (1.75)
360K - 4 mins 52 secs (2.21)
640K - 7 mins 48 secs (3.48)

The figure in brackets is the time as a multiple of the 800K's time. Interesting? I thought so. As an extra experiment I also tried an 800K to 800K copy with the same data but using the disc backup in 'DISK' and it took 4 minutes 48 seconds. Of course I couldn't use this for two different formats. As not everyone labelled their disc with its format, file to file copying was the only method guaranteed to work for all discs.

I suppose if I'd been asked beforehand I would have forecast that 360K would be the slowest, if not by very much. Since the copy was from a RAM disc that accounts for very little time, in fact just about 4 seconds for 200K. The remainder of the time is attributable to the floppy disc. Regardless of the RAM disc's contribution, the differences can be explained only by the different formats.

It really puts things in perspective when you realise it would still be quicker to first re-format 640K discs to 800K and then to copy them. For this reason I didn't time a 640 to 640 file copy. I leave it to you if you want to know; I'm happy to remain ignorant. I know why the 640K time is so poor, but I'd never thought about it before. The only 640K DOS disc I possess is my boot disc.

So why is 640K so slow? The answer is that it's the only 512 disc format that uses the ADFS ROM in the Beeb. Whatever the strengths of Acorn's ADFS, speed isn't one of them. A while ago I disassembled 6502.SYS and was interested to see that no IBM or CP/M formats go anywhere near ADFS - obvious really, but as I've said, it's one of those things you never think about until there's a reason.

You'll find the more files on a 640K disc the worse performance becomes. With only a dozen files it's not too bad, with 60 it's abysmal, with a couple of hundred I don't want to know. This is due to the way the (DOS) directories and files are managed in ADFS. ADFS sees the DOS disc as a single file, so reading or updating of files or directories is done (by DOS) using byte access only. This is the way a hard disc is handled too, but with a Winchester's speed it hardly matters; with floppies it does.

You can try these test yourself if you're curious and you've some spare time. You may get slightly different results, but they'll be close - I

don't use Acorn's ADFS because it's so slow. For example, I just loaded Acorn's ADFS into sideways RAM and booted DOS from floppy. The result including executing the 'AUTOEXEC' file was 1 minute 3 seconds for Acorn and 38 seconds using my normal ADFS. 'Acorn formatted' 640K copy times are longer too, but for all the non-ADFS formats the results should be about the same.

What does all this lead to? Two things. First, if you habitually use 640K discs (or the other sizes to a lesser degree) do yourself a big favour and convert your discs to 800K immediately. Not only will you find you have a number of spare discs, DOS applications and files will load very much faster. The second point? If you should have cause to send me a disc, please - only 800K in future.

PROBLEM SOLVER UPDATE

Thanks to everyone who contributed to this. Overall the view is rather mixed, some are happy with it, some not so. This leaves us more or less where we were in the last Forum. I know that at the rate of one Forum per month, developments appear slowly, so here's a progress update.

I wrote to Shibumi Soft again on June 10th and received a reply yesterday, July 4th. Further correspondence is needed, but with luck there'll be something more final in the next Forum. My advice for now is to hold on, but if you can't wait bear in mind the points most commonly mentioned by Forum readers and (mostly) confirmed by Shibumi Soft:

1. It runs better with DOS+ 1.2, and is decidedly delicate with 2.1 in all machines.

2. Use it only with applications that don't work without it. There is a list, not supplied as standard, so you're left with trial and error for problem packages. It can definitely cause problems with programs that do run correctly otherwise.

3. One or two packages (Elite) do run with it, but take about four minutes to start after loading. That's long enough to think it has failed or hung, perhaps it hasn't. Go and make a cup of coffee then see how it's getting on afterwards.

4. It does not work with the PC+, with either DOS+ 1.2 or 2.1.

5. It's copy protected, therefore you can't run it from a hard disc, nor can you make a back up copy.

6. The 'menu' mentioned in the documentation doesn't exist. (They say it's invisible!) Operation is from the DOS prompt and is not confirmed, nor can current settings be interrogated or displayed. This is perhaps the worst point; you're left entirely in the dark even if everything is alright.

7. I still can't get it to run correctly, even with 1.2 (Shibumi Soft say four others are in the same position and they don't know why).

And Finally...

Roger Gelder (who's also had no luck with Problem Solver) writes asking if anyone has information on graphics packages for the 512. He's heard that TURBO CAD V.1 is OK and knows of Ventura, but naturally is mostly impressed by its price. Information please, (not on GEM) if you use a graphics package.

MS DOS UTILITY

We have included an MS DOS utility by Bernard Hill on this month's disc called FIND.EXE. This will locate a specified string within an ASCII file by listing all lines containing the given string. A documentation file is also included on the disc.

The files will need to be transferred to the Master 512 using:

```
A:
GETFILE :1.FINDEXE FIND.EXE /DISC
GETFILE :1.HOWUSE HOWUSE.DOC /DISC
```

The documentation file also describes a second MS DOS utility which we expect to include on next month's magazine disc.

Spin a Disc (5)

David Spencer continues his series on disc systems with a look at the way DFS and ADFS use the disc space.

So far, we have concentrated on the way the disc controller lays out the fields on a disc. This month we will turn our attention towards the filing systems, and show how they make use of the disc space when storing files.

START AT THE BOTTOM

The way in which DFS uses disc space is much simpler than the method used by ADFS, so we will start with the former. A DFS disc has a directory structure which is essentially flat. This means that while files can be in different directories, all these directories appear together in a single catalogue - there is no concept of one directory being contained within another. Because of this very simple structure, the only information DFS needs to store on disc (apart from the files themselves) is a catalogue which contains the information for each file, including where it can be found on disc. From this, DFS can locate any file, and also deduce how much free space is available on the disc, and where it is associated.

The DFS catalogue takes up the first two sectors on track zero of each disc (or each side for a double sided disc). Table 1 shows the format of these two sectors. The first eight bytes of each sector contain information which is common to the disc (or the side of the disc) as a whole.

The disc title is as set using *TITLE, and if it is less than twelve characters long then the extra bytes are filled with zeros. The Master sequence number is the number which appears after the title when the disc is catalogued. This indicates how many times the disc has been written to since it was formatted. It is stored in a format known as Binary Coded Decimal (BCD) which means that the byte is split into two four-bit nibbles, each holding a digit of the number in binary. For example, the number 68 would be represented by the BCD value 0110 1000. The next byte holds the number of catalogue entries (the number of files on the disc). This is multiplied by eight so that it can act as an index into the rest of the catalogue, as we will see shortly. The size of the disc is represented by a ten bit number which holds the total number of sectors on the disc. As each sector is 256 bytes long, this is simply the disc size (in K) times four. Therefore for a 200K disc, the size in sectors is 800 (&320). The final piece of information held in the catalogue header is the boot option, as set by *OPT 4.

The remainder of the two catalogue sectors are split into 31 entries of sixteen bytes - one for each possible file. Each entry is split between eight bytes in the first sector, and eight in the second. Therefore, the first file's entry occupies bytes 8-15 of both sectors, the second bytes 16-23 and so on. For each entry, the first seven bytes contain the file's name, which is padded with spaces if it is less than seven characters. The eighth byte contains the directory letter, and also a bit indicating whether the file is locked against deletion or not. This is indicated by setting bit seven of the byte. Therefore, for a locked file in

directory '$', the eighth byte would contain ASC"$" + &80 which is &A4.

The eight bytes of each entry contained in sector 1 contain the load and execution address of the file, its length, and the sector number at which it starts. The exact format is shown in table 1.

Sector 0

Bytes	Contents
0-7	First eight bytes of title
8-14	First filename
15	First directory character and lock bit

Bytes 8-15 are repeated for up to 31 entries

Sector 1

Bytes	Contents
0-3	Last four bytes of title
4	Master sequence number
5	Number of catalogue entries multiplied by eight
6	Bits 0-1: high two bits number of sectors on disc
7	Bits 4-5: *OPT4 boot option
8	Bits 0-7 of load address
9	Bits 8-15 of load address
10	Bits 0-7 of exec address
11	Bits 8-15 of exec address
12	Bits 0-7 of file's length
13	Bits 8-15 of file's length
14	Bits 0-1: Bits 8-9 of the start sector of the file
	Bits 2-3: Bits 16-17 of the file's load address
	Bits 4-5: Bits 16-17 of the file's length
	Bits 6-7: Bits 16-17 of the file's execution address
15	Bits 0-7 of the files start sector

Bytes 8-15 are repeated for up to 31 entries

Table 1. The DFS catalogue structure

You might expect the catalogue to be stored in alphabetical order of filenames, as this is how it appears when *CAT is used. However, this is not the case and instead it is stored in the order in which the files are stored on the disc, with the last file first. The reason for this is that it is then much easier for the DFS to work out how many free sectors are on the disc, and how they are distributed. Therefore, whenever an entry is added to, or removed from, the catalogue, it is re-ordered so as to keep the entries in the correct sequence.

One thing to note is that the load and execute addresses are only allocated 18 bits, while internally 32-bit numbers are used. Furthermore, it is usual for addresses of the form FFFFxxxx to refer to the Beeb itself, and any other address to refer to the second processor. With the 18-bit addresses of the DFS, the top sixteen bits are reduced down to just two, and addresses of the form 3xxxx refer to the Beeb, and any other to the second processor.

ADFS DISCS

With ADFS discs, the situation is a different kettle of fish as they say. ADFS supports a hierarchical directory which means that a directory can contain a number of sub-directories, which in turn can contain further sub-directories ad infinitum (well up to the limit of disc space). Therefore it is not possible to have a single disc catalogue stored in a fixed place on the disc. Instead, each directory is stored on the disc in much the same way as a normal file, the only exception to this being the so-called root directory. This is the top level directory, and is always stored in sectors two to six of the disc. The whereabouts of all the other directories is determined by their entry in their parent's directory. In other words, a sub-directory can be thought of as a file, the contents of which are the sub-directory.

Table 2 shows the contents of an ADFS directory. You will see that all the space from byte 5 to 1226 contains the details of up to 47 entries. Unlike with DFS, the load and execution addresses are the full 32 bits long. The master sequence number for the directory (stored in bytes 0 and 1274) is similar to that used under DFS, and the

master sequence number stored for each entry is an indication of the order in which the files were written. The four bytes which spell 'Hugo' appear at the start and end of the directory, and these are used to indicate that this is indeed a directory. It is when these are corrupted that a Broken Directory error occurs. Incidentally, Hugo comes from Hugo Tyson who wrote the original version of ADFS. The file attributes (R, W, L etc.) are coded into the file's name, by setting or clearing bit 7 of the first five characters. The meaning of these bits is:

```
1st char - Bit 7 set => R attribute set
2nd char - Bit 7 set => W attribute set
3rd char - Bit 7 set => L attribute set
4th char - Bit 7 set => object is a
           directory (else it is a file)
5th char - Bit 7 set => E attribute set
```

If the name is less than ten bytes then it is terminated with a carriage return, and the remaining bytes are ignored. The same encoding applies to the directory name (bytes 1228-1237) which is the same as the name of the directory in its parent. This is different to the directory title (bytes 1241-1259) which is as set with *TITLE. The only other entry is the start sector of the parent directory held in bytes 1238-1240. This is used when a '^' appears in a pathname, meaning the parent directory. For the root directory, its parent is itself, so *CAT $.^ will in fact merely catalogue the root.

THE FREE SPACE MAP

Another consequence of having multiple catalogues is that it is no longer possible to locate the free space on a disc by examining the catalogue. (Well it is, but it would be a very lengthy process because the catalogue for each directory would need to be examined.) Instead, ADFS maintains a free space map (FS map) which holds details of all the areas of free space on the disc. This map is stored on the first two sectors of the disc (sectors 0 and 1 of track 0), and is read into memory when the disc is *MOUNTed. This allows ADFS to access it quickly without having to re-read the disc every time. However, each time a file is saved the free space map is updated and this is written back to the disc so that it is in a consistent state.

Bytes	Contents
0	Directory sequence number
1-4	The string 'Hugo'
5-14	Filename of the first file
15-18	Load address of first file
19-22	Exec address of first file
23-26	Length of first file
27-29	Start sector of first file
30	Sequence number of first file
Bytes 5 to 30 are repeated for up to 47 entries	
1227	Set to zero
1228-1237	Directory name
1238-1240	Start sector of parent
1241-1259	Directory title
1260-1273	Reserved (set to zero)
1274	Directory sequence number (Must be same as byte 0)
1275-1278	The string 'Hugo'
1279	Set to zero
If there are less than 47 entries, then the first byte after the last entry is set to zero to indicate this.	

Table 2. The ADFS directory format

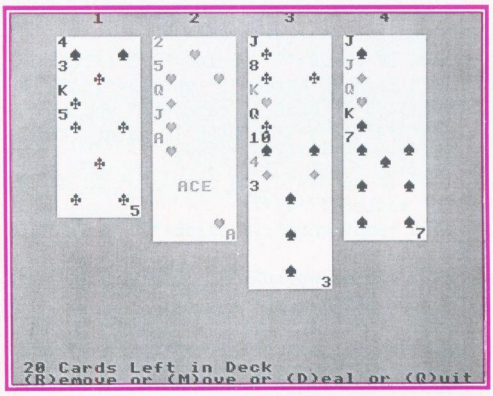
The format of the free space map is relatively straightforward. The first 246 bytes of sector 0 hold 82 three-byte entries, these giving the start sector for each area of free space on the disc. The corresponding 246 bytes of sector 1 hold the lengths of the free spaces. In practice there are unlikely to be as many as 82 separate chunks of free space, and therefore only a portion of the entries will be used. Of the remaining bytes, bytes 252 to 254 of sector 0 hold the total number of sectors on the disc. For a 40 track single sided disc this will be &280, while for an 80 track double sided disc it is &A00. Byte 254 of sector 1 holds a pointer to the end of the

Continued on page 58

♥ ♠ Aces High ♣ ♦

Geoff Steeper describes a delightfully simple game of patience known as Aces High.

This is a form of patience often called Aces High and the rules are quite simple. The object of the game is to remove all the cards from the table, and finish with the Aces at the head of each column.



Just type the program in and save it before you start to play. The cards are dealt four at a time one to each of four columns. A card can only be (R)emoved if there is a card of the same suit and a higher denomination at the bottom of one of the other columns. Aces are the highest value followed by Kings, Queens, Jacks and Tens, with the other cards in descending order. If one column is empty then any card from the bottom of another column can be (M)oved into the empty space. This is important because it can give access to other cards that can then be removed. When the cards at the bottom of each column are of different suits then four more cards must be (D)ealt.

This process continues until all the cards have been dealt. If you get all the Aces to the top and there are cards still left in the deck, you must continue until all the cards have been dealt. You then must continue to make legal moves until only the four Aces remain, one in each column, the position known as Aces High.

Although it may sound complicated, this is a delightfully simple yet addictive game of patience.

```
   10 REM Program ACES HIGH
   20 REM Version B0.6
   30 REM Author  Geoff Steeper
   40 REM BEEBUG  August/September 1989
   50 REM Program subject to copyright
   60 :
  100 ON ERROR MODE7:REPORT:PRINT" at li
ne ";ERL:END
  110 DIMcard$(4,13):DIMsuit$(4,13)
  120 PROCdefine:MODE1
  130 REPEAT:PROCinit
  140 REPEAT
  150 IF deck%=0 VDU7:PRINTTAB(1,29)"You
 still have a Legal Move";:GOTO190
  160 PROCdeal
  170 COLOUR0:COLOUR130
  180 PRINTTAB(1,29);deck%;" Cards Left
in Deck "
  190 REPEAT:REPEAT
  200 OK=0:PROCreval
  210 IF row1%<4 row1%=2:card$(1,0)="W":
suit$(1,0)="W"
  220 IF row2%<4 row2%=2:card$(2,0)="X":
suit$(2,0)="X"
  230 IF row3%<4 row3%=2:card$(3,0)="Y":
suit$(3,0)="Y"
  240 IF row4%<4 row4%=2:card$(4,0)="Z":
suit$(4,0)="Z"
  250 COLOUR0:COLOUR130
  260 PRINTTAB(1,30)"(R)emove or (M)ove
or (D)eal or (Q)uit"
  265 Move$=CHR$(GET AND &DF)
  270 UNTIL Move$="R" OR Move$="M" OR Mo
ve$="D" OR Move$="Q":PRINTTAB(0,30)SPC39
  280 IF Move$="R" PROCrem
  290 IF Move$="M" PROCmove
  300 IF Move$="Q" CLS:GOTO130
  310 IF gone%=48 Move$="D"
  320 UNTIL Move$="D"
  330 UNTIL deck%=0 AND suit$(1,0)<>suit
$(2,0) AND suit$(1,0)<>suit$(3,0) AND su
it$(1,0)<>suit$(4,0) AND suit$(2,0)<>sui
t$(3,0) AND suit$(2,0)<>suit$(4,0) AND s
uit$(3,0)<>suit$(4,0)
  340 PROCfinish
  350 UNTIL key$="N"
  360 MODE7
  370 END
  380 :
 1000 DEF PROCinit
 1010 deck%=52:gone%=0
 1020 row1%=2:row2%=2:row3%=2:row4%=2
 1030 VDU23,1,0;0;0;0;
```

```
 1035 VDU19,2,2,0,0,0
 1040 *FX11,0
 1050 COLOUR130:CLS:COLOUR0:PRINTTAB(12,
10);"A C E S  H I G H";TAB(12,17);"By  G
eoff Steeper";TAB(4,25)"Press 'SPACE' to
 Shuffle and Deal";
 1060 REPEAT UNTIL GET=32:CLS
 1070 PRINTTAB(7,0);"1";TAB(15,0);"2";TA
B(23,0);"3";TAB(31,0);"4"
 1080 shuffled$="AHACADAS2H2C2D2S3H3C3D3
S4H4C4D4S5H5C5D5S6H6C6D6S7H7C7D7S8H8C8D8
S9H9C9D9STHTCTDTSJHJCJDJSQHQCQDQSKHKCKDK
S"
 1090 PROCshuffle:PROCshuffle
 1100 ENDPROC
 1110 :
 1120 DEF PROCshuffle
 1130 deck$="":FOR shuffle%=51 TO 0 STEP
-1
 1140 random%=RND(shuffle%):deck$=deck$+
MID$(shuffled$,2*random%+1,2):shuffled$=
LEFT$(shuffled$,random%*2)+RIGHT$(shuffl
ed$,(shuffle%-random%)*2)
 1150 NEXTshuffle%:shuffled$=deck$
 1160 ENDPROC
 1170 :
 1180 DEF PROCdeal
 1190 col%=0
 1200 FOR col=1 TO 4
 1210 col%=col%+1
 1220 card$=LEFT$(deck$,1):deck$=RIGHT$(
deck$,LEN(deck$)-1):suit$=LEFT$(deck$,1)
:deck$=RIGHT$(deck$,LEN(deck$)-1)
 1230 xcoord%=8*col%-4
 1240 IF col%=1 row%=row1%:row1%=row1%+2
:card$(1,(row1%-2)/2)=card$:suit$(1,(row
1%-2)/2)=suit$
 1250 IF col%=2 row%=row2%:row2%=row2%+2
:card$(2,(row2%-2)/2)=card$:suit$(2,(row
2%-2)/2)=suit$
 1260 IF col%=3 row%=row3%:row3%=row3%+2
:card$(3,(row3%-2)/2)=card$:suit$(3,(row
3%-2)/2)=suit$
 1270 IF col%=4 row%=row4%:row4%=row4%+2
:card$(4,(row4%-2)/2)=card$:suit$(4,(row
4%-2)/2)=suit$
 1280 PROCdisplay
 1290 deck%=deck%-1
 1300 NEXT col
 1310 ENDPROC
 1320 :
 1330 DEF PROCdefine
 1340 VDU23,224,0,8,28,62,127,62,28,8
 1350 VDU23,225,0,54,127,127,127,62,28,8
 1360 VDU23,226,8,28,62,127,127,127,28,6
2
 1370 VDU23,227,8,28,28,107,127,107,8,28
 1380 ENDPROC
 1390 :
 1400 DEF PROCdisplay
 1410 LOCAL temp,temp$,whole$,part,line$
,line,count,char
 1420 IF card$="A" whole$="7*45*7" ELSE
IF card$="2" whole$="9*41*9" ELSE IF car
d$="3" whole$="9*20*20*9" ELSE IF card$=
"4" whole$="7*3*37*3*7" ELSE IF card$="5
" whole$="7*3*18*18*3*7" ELSE IF card$="
6" whole$="7*3*16*3*16*3*7"
 1430 IF card$="7" whole$="7*3*4*11*3*16
*3*7" ELSE IF card$="8" whole$="7*3*4*11
*3*11*4*3*7" ELSE IF card$="9" whole$="7
*3*9*3*4*4*3*9*3*7" ELSE IF card$="T"who
le$="6*3*4*4*3*9*3*4*4*3*6"
 1440 IF card$="J" OR card$="Q" OR card$
="K" whole$="7*45*7"
 1450 temp$=card$:IF card$="T" temp$="10
"
 1460 whole$=whole$+"*":temp%=0:FOR coun
t%=1 TO LEN(whole$)-1
 1470 IF MID$(whole$,count%,1)<>"*" AND
MID$(whole$,count%+1,1)<>"*" temp$=temp$
+STRING$(VAL(MID$(whole$,count%,1))*10,"
 ")
 1480 IF MID$(whole$,count%,1)<>"*" AND
MID$(whole$,count%+1,1)="*" temp$=temp$+
STRING$(VAL(MID$(whole$,count%,1))," ")
 1490 IF MID$(whole$,count%,1)="*" temp$
=temp$+"*"
 1500 NEXT count%:IF card$="T" whole$=te
mp$+"10"
 1510 IF card$<>"T" whole$=temp$+card$
 1520 COLOUR131:COLOUR1:IF suit$="D" cha
r%=224 ELSE IF suit$="H" char%=225 ELSE
IF suit$="S" char%=226:COLOUR0 ELSE IF s
uit$="C" char%=227:COLOUR0
 1530 FOR line%=0 TO 8:line$=MID$(whole$
,(line%*7)+1,7)
 1540 FOR part=1 TO 7:IF MID$(line$,part
,1)="*" line$=LEFT$(line$,part-1)+CHR$(c
har%)+RIGHT$(line$,7-part)
 1550 NEXT part
 1552 PRINTTAB(xcoord%,row%+line%);line$
 1555 NEXT line%
 1560 IF card$="A" PRINTTAB(xcoord%+2,ro
w%+4);"ACE"
 1570 card$((xcoord%+4)/8,0)=card$
 1575 suit$((xcoord%+4)/8,0)=suit$
 1580 ENDPROC
 1590 :
 1600 DEF PROCmove
 1610 REPEAT:PRINTTAB(1,30);"Move Card f
rom Col.No.?";
```

```
 1620 col%=GET-48
 1630 UNTIL col%=1 OR col%=2 OR col%=3 O
R col%=4
 1640 PRINTTAB(23,30);col%
 1650 I%=1:REPEAT:move%=I%
 1660 PROCcheckmove:I%=I%+1
 1670 UNTIL OK OR I%>4
 1680 IF OK=0 VDU7:ENDPROC
 1690 PROCrem2
 1700 card$=card$(col%,(row%-2)/2):suit$
=suit$(col%,(row%-2)/2)
 1710 card$(move%,1)=card$:suit$(move%,1
)=suit$
 1720 IFmove%=1 xcoord%=4:row1%=row1%+2
 1730 IFmove%=2 xcoord%=12:row2%=row2%+2
 1740 IFmove%=3 xcoord%=20:row3%=row3%+2
 1750 IFmove%=4 xcoord%=28:row4%=row4%+2
 1760 row%=2:PROCdisplay
 1770 ENDPROC
 1780 :
 1790 DEF PROCrem
 1800 COLOUR0:COLOUR130
 1810 REPEAT:PRINTTAB(1,30);"Remove Card
 from Col.No.?";
 1820 col%=GET-48
 1830 UNTIL col%>=1 AND col%<=4
 1840 PROCcheckrem
 1850 IF OK=0 VDU7:ENDPROC
 1860 PROCrem2
 1870 gone%=gone%+1
 1880 ENDPROC
 1890 :
 1900 DEF PROCrem2
 1910 IF col%=1 PROCremove(4,row1%):row%
=row1%:row1%=row1%-2
 1920 IF col%=2 PROCremove(12,row2%):row
%=row2%:row2%=row2%-2
 1930 IF col%=3 PROCremove(20,row3%):row
%=row3%:row3%=row3%-2
 1940 IF col%=4 PROCremove(28,row4%):row
%=row4%:row4%=row4%-2
 1950 ENDPROC
 1960 :
 1970 DEF PROCremove(X%,Y%)
 1980 IF Y%<4 Y%=4
 1990 card$=card$(col%,(Y%-4)/2):suit$=s
uit$(col%,(Y%-4)/2)
 2000 FOR Step%=0 TO 8
 2010 PRINTTAB(X%,Y%-2+Step%) SPC(8)
 2020 NEXT
 2030 IF Y%<=4 ENDPROC
 2040 xcoord%=X%:row%=Y%-4:PROCdisplay
 2050 ENDPROC
 2060 :
 2070 DEF PROCcheckrem
 2080 IF col%=1 AND (suit$(1,0)=suit$(2,
0) OR suit$(1,0)=suit$(3,0) OR suit$(1,0
)=suit$(4,0)) PROCcheckrem2
 2090 IF col%=2 AND (suit$(2,0)=suit$(1,
0) OR suit$(2,0)=suit$(3,0) OR suit$(2,0
)=suit$(4,0)) PROCcheckrem2
 2100 IF col%=3 AND (suit$(3,0)=suit$(1,
0) OR suit$(3,0)=suit$(2,0) OR suit$(3,0
)=suit$(4,0)) PROCcheckrem2
 2110 IF col%=4 AND (suit$(4,0)=suit$(1,
0) OR suit$(4,0)=suit$(2,0) OR suit$(4,0
)=suit$(3,0)) PROCcheckrem2
 2120 ENDPROC
 2130 :
 2140 DEF PROCcheckmove
 2150 IF move%=1 AND card$(1,0)="W" OK=1
 2160 IF move%=2 AND card$(2,0)="X" OK=1
 2170 IF move%=3 AND card$(3,0)="Y" OK=1
 2180 IF move%=4 AND card$(4,0)="Z" OK=1
 2190 IF card$(col%,0)="W" OR card$(col%
,0)="X" OR card$(col%,0)="Y" OR card$(co
l%,0)="Z" OK=0
 2200 ENDPROC
 2210 :
 2220 DEF PROCcheckrem2
 2230 IF VAL(card$(col%,0)) < VAL(card$(
1,0)) AND suit$(col%,0)=suit$(1,0) OK=1
 2240 IF VAL(card$(col%,0)) < VAL(card$(
2,0)) AND suit$(col%,0)=suit$(2,0) OK=1
 2250 IF VAL(card$(col%,0)) < VAL(card$(
3,0)) AND suit$(col%,0)=suit$(3,0) OK=1
 2260 IF VAL(card$(col%,0)) < VAL(card$(
4,0)) AND suit$(col%,0)=suit$(4,0) OK=1
 2270 ENDPROC
 2280 :
 2290 DEF PROCreval
 2300 FOR C=1 TO 4
 2310 IF card$(C,0)="A" card$(C,0)="14"
 2320 IF card$(C,0)="K" card$(C,0)="13"
 2330 IF card$(C,0)="Q" card$(C,0)="12"
 2340 IF card$(C,0)="J" card$(C,0)="11"
 2350 IF card$(C,0)="T" card$(C,0)="10"
 2360 NEXT C
 2370 ENDPROC
 2380 :
 2390 DEF PROCfinish
 2400 IF gone%=48 PRINTTAB(0,29);SPC119;
TAB(12,20)"ACES ARE HIGH";TAB(14,22)"WEL
L DONE":VDU7:GOTO2420
 2410 PRINTTAB(1,29);"HARD LUCK - YOU HA
VE RUN OUT OF MOVES"
 2420 VDU7:PRINTTAB(5,31);"Do you want a
nother go (Y/N)?";
 2430 REPEAT:key$=CHR$(GET AND &DF):UNTI
L key$="Y" OR key$="N"
 2440 ENDPROC
```

```
 1160 ParBlk!12=HelpMess
 1170 ParBlk?14=L%
 1180 ParBlk?15=135
 1190 ParBlk?16=8
 1200 $PromptMessage=P$
 1210 $InputBuffer=I$
 1220 $HelpMess=CHR$(40-LENH$/2)+CHR$23+
H$
 1230 X%=ParBlk MOD256:Y%=ParBlk DIV256:
CALLMc
 1240 Return%=(ParBlk?3)AND254:EditMode%
=(ParBlk?3)AND1
 1250 =$InputBuffer
 1260 :
 1270 DEF PROCDemo
 1280 DIM ParBlk 20,PromptMessage 80,Inp
utBuffer 255,HelpMess 255
 1290 READ N%
 1300 DIM P$(N%),PX%(N%),PY%(N%),Input$(
N%),IX%(N%),IY%(N%),Len%(N%),H$(N%)
 1310 DIM Flag1% N%,Flag2% N%
 1320 DIM ed$(1):ed$(0)="Insert    ":ed$(
1)="Overwrite"
 1330 FOR i%=0 TO N%-1:READ P$(i%),PX%(i
%),PY%(i%),Input$(i%),IX%(i%),IY%(i%),Le
n%(i%),H$(i%),Flag1%?i%,Flag2%?i%
 1340 Dummy$=FNEditLine(P$(i%),PX%(i%),P
Y%(i%),Input$(i%),IX%(i%),IY%(i%),Len%(i
%),H$(i%),Flag1%?i%,Flag2%?i%,TRUE)
 1350 NEXT
 1360 i%=0:PROCMode
 1370 REPEAT:ParBlk?18=0
 1380 REPEAT:Input$(i%)=FNEditLine(P$(i%
),PX%(i%),PY%(i%),Input$(i%),IX%(i%),IY%
(i%),Len%(i%),H$(i%),Flag1%?i%,Flag2%?i%
,FALSE)
 1390 IF Return%=2 PROCMode
 1400 IF Return%=4 PROCHelp
 1410 UNTIL Return%>15
 1420 IF Return%=64 i%=(i%-1) MODN%-N%*(
i%=0)
 1430 IF Return%=32 i%=(i%+1) MODN%
 1440 UNTIL Return%=16 OR Return%=128
 1450 ENDPROC
 1460 :
 1470 DEF PROCHelp
 1480 TIME=0:REPEAT UNTIL TIME>250
 1490 PRINTTAB(0,23)SPC(80);:ENDPROC
 1500 :
 1510 DEF PROCMode
 1520 PRINTTAB(46,1)"Current editing mod
e is ";ed$(EditMode%)
 1530 ENDPROC
 1540 :
 1550 DATA19
 1560 DATA"Surname",1,1,"",15,1,25,"Ente
r your surname, e.g. PENNINGTON-SMYTHE",
&A0,&45
 1570 DATA"Forename(s)",1,3,"",15,3,40,"
Enter your forename(s)/christian names,
e.g. Harvey Smedley",&A0,&44
 1580 DATA"House name",1,5,"",15,5,23,"E
nter the name of your house, if any, e.g
. Dunroamin",&A0,&44
 1590 DATA"House number",1,7,"",15,7,4,"
Enter the number of your house, if it ha
s one",&A0,&44
 1600 DATA"Street",1,9,"",15,9,23,"Enter
 the name of the street",&A0,&44
 1610 DATA"District",1,11,"",15,11,23,"E
nter the name of the district",&A0,&44
 1620 DATA"Town",40,5,"",54,5,25,"Enter
the name of the town",&A0,&44
 1630 DATA"County",40,7,"",54,7,25,"Ente
r the name of the county",&A0,&45
 1640 DATA"Postcode",40,9,"",54,9,9,"Ent
er your postcode, e.g. LU8 5TG",&A0,&45
 1650 DATA"Type any digit",54,11,"",69,1
1,1,"Digits are 0, 1, 2, 3, 4, 5, 6, 7,
8 or 9",&A9,&4A
 1660 DATA"How many CSEs do you have?",1
,13,"",32,13,2,"Enter the number of CSE
passes you have",&A8,&4A
 1670 DATA"How many O-levels do you have
?",1,15,"",32,15,2,"Enter the number of
O-level passes you have",&A8,&4A
 1680 DATA"How many A-levels do you have
?",1,17,"",32,17,2,"Enter the number of
A-level passes you have",&A8,&4A
 1690 DATA"Do you like football? (Y/N)",
40,13,"",78,13,1,"Well? Do you?",&A2,&44
 1700 DATA"Are you a yuppie? (Y/N)",40,1
5,"",78,15,1,"Seek help if you answer 'Y
' to this question",&A2,&44
 1710 DATA"Are you an estate agent? (Y/N
)",40,17,"",78,17,1,"Seek help if you an
swer 'Y' to this question",&A2,&44
 1720 DATA"Would you mind answering extr
emely personal questions? (Y/N) ",1,19,"
",-1,-1,1,"Are you going to be a party-p
ooper, or a really swinging dude?",&A2,&
44
 1730 DATA"Please enter the top secret p
assword:",1,21,"",-1,-1,15,"Only YOU kno
w the password -- I can't help you",&A0,
&64
 1740 DATA"Press Copy to finish: ",57,21
,"",-1,-1,1,"It's quite simple, press th
e Copy key when you've finished entering
 data",&C4,&44
```

RISC USER

The Archimedes Support Group

Our Risc User magazine is now in its second volume and is enjoying the largest circulation of any magazine devoted to the Archimedes. The list of members seeking support from the Risc User group is growing steadily and as well as private individuals includes schools, colleges, universities and industry and government establishments.

Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines. A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer, particularly at this time while documentation on the Archimedes and RISC OS is still limited.

Here are just some of the topics covered in more recent issues of RISC User:

RISC USER DESKTOP DIARY
A multi-tasking Desktop diary facility.

THE ARM 3 RISC PROCESSOR
An in-depth look by the designer of this exciting development.

ACORN DESKTOP PUBLISHER

MASTERING THE WIMP
A major new series for beginners to the WIMP programming environment.

A MULTI-TASKING NOTE-PAD
A comprehensive RISC OS multi-tasking application which offers a multi-page, multi-file notepad with editing and printing facilities.

INTO THE ARC
Series for newcomers to the Archimedes.

RISC USER TOOLBOX
A comprehensive toolbox module for the Archimedes.

DESKTOP BASIC HANDLER
A Desktop utility to convert basic programs into their text equivalents and vice versa.

MOUSE MENU
A general purpose WIMP based menu, which is fully RISC OS compatible.

PARTIAL RENUMBER UTILITY
A machine code utility for renumbering parts of a Basic program.

THE R140 UNIX WORKSTATION
A look at Acorn's entry into the business market.

ARM CODE SINGLE STEPPER
An ARM code debugging aid.

Don't delay - Phone your instructions now on (0727) 40303

As a member of BEEBUG you may extend your subscription to include RISC User for only £8.50 (overseas see below).

Name:.....
Memb No:.....
Address: .....
.....
.....
.....

SUBSCRIPTION DETAILS

Destination	Additional Cost
UK,BFPO &Ch Is	£ 8.50
Rest of Europe and Eire	£13.00
Middle East	£15.00
Americas and Africa	£17.00
Elsewhere	£19.00

I wish to receive both BEEBUG and ***RISC User****.* *I enclose a cheque for £ ..... or alternatively*

I authorise you to debit myACCESS/Visa/Connect account: _____/ _____/ _____/ _____/

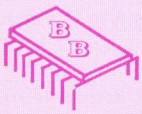
Signed: ..... *Expiry Date:* _____/ _____/

Send to: RISC User, 117 Hatfield Road, St Albans, Herts AL1 4JS, or telephone (0727) 40303

```
  690 !F%=!F% EOR-1:NEXT
  700 ON option% GOSUB 2760,2880,2930,30
10,3130,3240,3310,3380
  710 UNTIL option%=9
  730 MODE7:*FX4
  740 *FX220,27
  750 END
  760 :
  770 *|" *** STORE CHAR *** "
  780 :
  790 C%=FNget(FALSE)
  800 IF C%=TRUE THEN 850
  810 PROCval:M%=FNaddr1(y%,x%)
  820 FOR F%=0 TO 7:F%?M%=val%(F%):NEXT
  830 M%=FNaddr2(C%)
  840 FOR F%=0 TO 7:M%?F%=val%(F%):NEXT
  850 PROCreset
  860 RETURN
  870 :
  880 *|" *** PICKUP *** "
  890 :
  900 C%=FNget(FALSE):IF C%=TRUE THEN 99
0
  910 M%=FNaddr2(C%)
  920 FOR y%=1 TO 8
  930 V%=?(M%+(y%-1)):B%=128
  940 FOR x%=1 TO 8
  950 IF V%>=B% V%=V%-B%:G%=TRUE ELSE G%
=FALSE
  960 B%=B%/2:gr%(x%,y%)=G%
  970 PROCcur(FALSE,FALSE):PROCplot(-G%)
  980 NEXT x%,y%
  990 PROCreset
 1000 RETURN
 1010 :
 3860 DEF FNval
 3870 V%=0:B%=128
 3880 FOR N%=0 TO 7
 3890 G%=N%?ws1%
 3900 IF G% THEN V%=V%+B%
 3910 B%=B%/2
 3920 NEXT
 3930 =V%
 3940 :
 4040 DEF FNdec(p0%,p1%)=LEFT$(STRING$(p
0%,"0"),p0%-LENSTR$p1%)+STR$p1%
 4050 :
 4060 DEF FNget(p%)
 4070 LOCAL E%
 4080 PROCcl_win:PROCescF
 4090 SOUND2,-14,150,4
 4100 PRINT TAB(14,5)"Select"TAB(12,6)"C
haracter";:E%=0:C%=32
 4110 xx%=x%:yy%=y%:x%=25:y%=2
 4120 PROCcur(FALSE,TRUE)
 4130 REPEAT
 4140 IF INKEY-26 THEN PROClt(TRUE)
 4150 IF INKEY-122 THEN PROCrt(TRUE)
 4160 IF INKEY-58 THEN PROCup(TRUE)
 4170 IF INKEY-42 THEN PROCdn(TRUE)
 4180 PRINT TAB(13,7)FNdec(3,C%)" - ";
 4190 IF C%>126 THEN VDU32 ELSE VDU C%
 4200 IF INKEY-113 THEN E%=TRUE
 4210 IF p% AND INKEY-99 THEN E%=1:C%=1
 4220 UNTIL INKEY-74 OR E%
 4230 PROCcur(FALSE,TRUE)
 4240 IF E%=TRUE THEN C%=TRUE
 4250 =C%
 4260 :
 4270 DEF FNkey
 4280 R%=FALSE
 4290 IF INKEY-33 THEN R%=1
 4300 IF INKEY-114 THEN R%=2
 4310 IF INKEY-115 THEN R%=3
 4320 IF INKEY-116 THEN R%=4
 4330 IF INKEY-21 THEN R%=5
 4340 IF INKEY-117 THEN R%=6
 4350 IF INKEY-118 THEN R%=7
 4360 IF INKEY-23 THEN R%=8
 4370 IF INKEY-119 R%=9
 4380 =R%
 4390 :
 4400 DEF FNaddr1(p0%,p1%)=&5800+(p0%*&1
40)+(8*p1%)
 4410 :
 4420 DEF FNaddr2(p%)=(font%-&100)+8*p%
 4430 :
 4440 *|" *** PROCEDURES *** "
 4450 :
 4540 DEF PROCescF
 4550 esc%=FALSE:*FX220,17
 4560 ENDPROC
 4570 :
 4890 DEF PROCreset
 4900 x%=xx%:y%=yy%:SOUND3,-14,200,4
 4910 *FX21
 4920 PROCr_ins
 4930 ENDPROC
 4940 :
 4950 DEF PROCr_ins
 4960 PRINTw2$;:PROCinstr(i%):PROCcl_win
 4970 ENDPROC
 4980 :
 4990 DEF PROCcl_win
```

```
 5000 VDU26
 5010 FOR F%=5 TO 8:PRINT TAB(12,F%)SPC9
;:NEXT
 5020 ENDPROC
 5030 :
 5040 DEF PROCfont
 5050 C%=font%
 5060 FOR F%=2 TO 28 STEP 2
 5070 FOR N%=25 TO 37 STEP 2
 5080 M%=FNaddr1(F%,N%)
 5090 FOR L%=0 TO 1:!M%=!C%:M%=M%+4:C%=C
%+4:NEXT
 5100 NEXT N%,F%
 5110 ENDPROC
 5120 :
 5130 DEF PROCmove(p0%,p1%)
 5140 FOR F%=0 TO &2FC STEP 4
 5150 F%!p1%=F%!p0%:NEXT
 5160 ENDPROC
 5170 :
 5180 DEF PROCspc
 5190 PROCcl_win
 5200 PRINT TAB(3,27)"'SPACE' continues"
 5210 REPEAT G%=GET:UNTIL INKEY-99
 5220 PROCcl_win
 5230 ENDPROC
 5240 :
 5250 DEF PROCval
 5260 FOR F%=0 TO 7:FOR N%=0 TO 7
 5270 N%?ws1%=gr%(N%+1,F%+1):NEXT
 5280 val%(F%)=FNval
 5290 NEXT
 5300 ENDPROC
 5310 :
 5320 DEF PROCshift(p%)
 5330 ON p% GOTO 5340,5370,5450,5480
 5340 T%=gr%(1,y%)
 5350 FOR F%=1 TO7:gr%(F%,y%)=gr%(F%+1,y
%):NEXT
 5360 gr%(8,y%)=T%:GOTO 5400
 5370 T%=gr%(8,y%)
 5380 FOR F%=7 TO 1 STEP -1:gr%(F%+1,y%)
=gr%(F%,y%):NEXT
 5390 gr%(1,y%)=T%
 5400 xx%=x%
 5410 FOR x%=1 TO 8
 5420 PROCbit(NOTgr%(x%,y%))
 5430 NEXT
 5440 x%=xx%:GOTO 5540
 5450 T%=gr%(x%,1)
 5460 FOR F%=1 TO 7:gr%(x%,F%)=gr%(x%,F%
+1):NEXT
 5470 gr%(x%,8)=T%:GOTO 5510
 5480 T%=gr%(x%,8)
 5490 FOR F%=7 TO 1 STEP -1:gr%(x%,F%+1)
=gr%(x%,F%):NEXT
 5500 gr%(x%,1)=T%:yy%=y%
 5510 yy%=y%
 5520 FOR y%=1 TO 8:PROCbit(NOTgr%(x%,y%
)):PROCcur(FALSE,FALSE):NEXT
 5530 y%=yy%
 5540 ENDPROC
 5550 :
 5560 DEF PROCbit(p%)
 5570 PROCplot(p%+1)
 5580 REPEAT:gr%(x%,y%)=NOT(gr%(x%,y%)):
UNTIL gr%(x%,y%)=NOT(p%)
 5590 ENDPROC
 5600 :
 5610 DEF PROCplot(p1%)
 5620 GCOL0,p1%
 5630 PLOT69,520+(x%*4),971-(y%*4)
 5640 ENDPROC
 5650 :
 5660 DEF PROClt(p%)
 5670 IF sh% THEN PROCshift(1):GOTO 5750
 5680 PROCcur(FALSE,p%)
 5690 IF p% THEN 5720
 5700 x%=x%-1:IF x%=0 THEN x%=8
 5710 GOTO 5750
 5720 C%=C%-1:x%=x%-2
 5730 IF x%=23 THEN x%=37:C%=C%+7
 5740 PROCcur(FALSE,TRUE)
 5750 ENDPROC
 5760 :
 5770 DEF PROCrt(p%)
 5780 IF sh% THEN PROCshift(2):GOTO 5870
 5790 PROCcur(FALSE,p%)
 5800 IF p% THEN 5840
 5810 x%=x%+1
 5820 IF x%=9 THEN x%=1
 5830 GOTO 5870
 5840 C%=C%+1:x%=x%+2
 5850 IF x%=39 THEN x%=25:C%=C%-7
 5860 PROCcur(FALSE,TRUE)
 5870 ENDPROC
 5880 :
 5890 DEF PROCup(p%)
 5900 IF sh% THEN PROCshift(3):GOTO 5990
 5910 PROCcur(FALSE,p%)
 5920 IF p% THEN 5960
 5930 y%=y%-1
 5940 IF y%=0 THEN y%=8
 5950 GOTO 5990
```

```
 5960 C%=C%-7:y%=y%-2
 5970 IF y%=0 THEN y%=28:C%=C%+98
 5980 PROCcur(FALSE,TRUE)
 5990 ENDPROC
 6000 :
 6010 DEF PROCdn(p%)
 6020 IF sh% THEN PROCshift(4):GOTO 6110
 6030 PROCcur(FALSE,p%)
 6040 IF p% THEN 6080
 6050 y%=y%+1
 6060 IF y%=9 THEN y%=1
 6070 GOTO 6110
 6080 C%=C%+7:y%=y%+2
 6090 IF y%=30 THEN y%=2:C%=C%-98
 6100 PROCcur(FALSE,TRUE)
 6110 ENDPROC
 6120 :
 6130 DEF PROCcur(p0%,p1%)
 6140 IF p1% THEN 6190
 6150 C%=gr%(x%,y%):P%=FNaddr1(y%,x%)
 6160 IF C% THEN p2%=&BD8181FF:p3%=&8181
BDBD ELSE p2%=&818181FF:p3%=&81818181
 6170 IF p0% THEN p2%=&7E7E00FF:p3%=&7E7
E7E
 6180 !P%=p2%:!(P%+4)=p3%:GOTO 6210
 6190 M%=FNaddr1(y%,x%)
 6200 FOR F%=0 TO7:F%?M%=(F%?M% EOR&FF):
NEXT
 6210 ENDPROC
 6220 :
 6230 DEF PROCwndw(p1%,p2%,p3%,p4%)
 6240 !&70=p1%:!&72=p2%:!&74=p3%:!&76=p4
%
 6250 CALL box%
 6260 ENDPROC
 6270 :
 6280 DEF PROCinstr(p%)
 6290 IF p% THEN RESTORE 6690 ELSE RESTO
RE 6800
 6300 PRINT w1$;w0$
 6310 REPEAT READ R$:PRINT R$:UNTIL R$="
"
 6320 VDU26
 6330 IF NOT(p%) THEN PROCshade(FNaddr1(
15+op%,6))
 6340 ENDPROC
 6350 :
 6360 DEF PROCshade(P%)
 6370 FOR F%=0 TO 87 STEP 4:P%!F%=(P%!F%
) AND &AA55AA55:NEXT
 6380 ENDPROC
 6430 :
 6440 *|" *** ERROR TRAP *** "
 6450 :
 6460 ON ERROR OFF:VDU7
 6470 IF ?&24A<>?&24B OR ERR=254 THEN 66
00
 6480 IF ERR=17 AND esc% THEN 6630
 6490 VDU3,22,7,129
 6500 IF ERR=17 THEN PRINT"PROGRAM STOPP
ED!":GOTO 6570
 6510 PRINT"ERROR:";:REPORT:PRINT" at li
ne "ERL
 6520 *FX21
 6530 L$="LIST "+STR$ERL+CHR$6+CHR$13
 6540 VDU21
 6550 A%=138:X%=0
 6560 FOR F%=1 TO LEN(L$):Y%=ASC(MID$(L$
,F%,1)):CALL!&20A:NEXT
 6570 *FX4
 6580 *FX220,27
 6590 END
 6600 VDU3,28,1,26,21,12,12
 6610 REPORT:VDU10,13:PROCspc
 6620 PROCinstr(i%):GOTO 480
 6630 PROCescF:PROCr_ins
 6640 *FX154,8
 6650 GOTO 480
 6660 :
 6670 *| " *** DATA *** "
 6680 :
 6690 DATA f0 Store char.
 6700 DATA f1 Load char.
 6710 DATA f2 Invert window
 6720 DATA f3 Rotate window
 6730 DATA f4 Mirror window
 6740 DATA f5 Clear wndw/char
 6750 DATA f6 Print chars
 6760 DATA f7 Wipe whole font
 6770 DATA f8 Exit
 6780 DATA " "," "," "
 6790 DATA 'SPACE' for more,""
 6800 DATA SHIFT +
 6810 DATA f0 Select Disc
 6820 DATA f1 Select Tape
 6830 DATA f2 Catalogue
 6840 DATA f3 Load font
 6850 DATA f4 Save font
 6860 DATA f5 Store font
 6870 DATA f6 *Command
 6880 DATA f7 Restore font
 6890 DATA f8 Exit
 6900 DATA " "," "
 6910 DATA 'SPACE' Returns,""
```



Basic Booster

The Best of BEEBUG

The Basic Booster ROM from BEEBUG is a selection of useful utilities to 'boost ' the Basic in your computer and give more power to your programming.
Some of the utilities are enhanced versions of programs previously published in BEEBUG magazine, while others are totally new. For those with no spare ROM sockets, Basic Booster is also available on disc to load into Sideways RAM.

Partial Renumber

A very useful utility which renumbers a selected block of lines. Ideal for adding extra lines to the middle of a lengthy program.

SPECIAL OFFER! Order your Basic Booster before the 30 September and you can have it for £6.00 (members only)!

Super Squeeze

A program compressor which can remove REMs, blank lines, spaces, and compress variable names.

Textload and Textsave

Save and load a Basic program as text. Saves the hassle of using *SPOOL and *EXEC.

Program Lister

List any program (including Archimedes Basic) direct from a file. Formatting can be controlled using the same options as are available on the Archimedes, including splitting multi-statement lines, omitting line numbers and listing keywords in lower case.

Resequencer

Rearrange the lines in a Basic program. Any line, or block of lines, can be moved or copied to any place in the program. Line numbering is automatically adjusted as blocks are moved.

Smart Renumber

Renumber a program to a standard format with procedures starting at a particular line number.

Please rush me my Basic Booster ROM/Disc:

Basic Booster ROM *Code 1403A* **£6.00**
Basic Booster ROM *(non-members)* **£19.95**

Basic Booster Disc *Code 1402A* **£6.00**
Basic Booster Disc *(non-members)* **£18.95**

Name _____

Address _____

_____

Memership No _____

ROM/Disc *(delete as necessary)* £_____

Postage £ 0.60

Total £_____

I enclose a cheque for £_____ OR please debit my Access, Visa or Connect account,
Card No _____/_____/_____/_____ Expiry_____/____ Signed_____

Return to BEEBUG Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Telephone (0727) 40303.

```
1600 DEF PROCvline(x,y1,y2)
1610 LOCAL p
1620 FOR p=y1 TO y2
1630 PROCpixel(x,p)
1640 NEXT p
1650 ENDPROC
1660 :
1700 DEF PROChline(x1,x2,y)
1710 LOCAL p
1720 FOR p=x1 TO x2
1730 PROCpixel(p,y)
1740 NEXT p
1750 ENDPROC
1760 :
```

Again, both these procedures use the pixel co-ordinate system (0 to 79 horizontally, 0 to 74 vertically downwards). All of these procedures are illustrated by the demonstration program listed at the end. To run the demo, simply type in all the procedures and functions as listed above, together with the main program. Then save to disc before running.

NAME	CO-ORDINATES
PROCmsg1(x,y,c,text$)	0 to 39, 0 to 24
PROCmsg2(x,y,c,text$)	0 to 39, 0 to 24
PROCframe(x,y,w,h)	0 to 39, 0 to 24
PROCforeground(x,y1,y2,c)	0 to 39, 0 to 24
PROCbackground(x1,x2,y,c)	0 to 39, 0 to 24
PROCpixel(x,y)	0 to 79, 0 to 74
PROCvline(x,y1,y2)	0 to 79, 0 to 74
PROChline(x1,x2,y)	0 to 79, 0 to 74

Table 1. List of Teletext procedures

As I said before, all the procedures and the demo program are intended as examples of what can be achieved. When using such procedures do be aware of all the hidden control codes in use, as the positions of these can easily corrupt otherwise good-looking displays. Pre-planning is the answer here.

Finally, we have included an additional program on the magazine disc for this issue, a full Teletext Editor. This very comprehensive program is written in machine code, and is unsuitable for publication in the magazine itself. Some brief notes on its operation are included separately in the magazine.

That concludes our present coverage of the Teletext mode. I hope you have found it instructive. Next month we shall be looking at something quite different.

```
   10 REM Program TeleProcs
   20 REM Version B1.3
   30 REM Author  Mike Williams
   40 REM BEEBUG August/September 1989
   50 REM Program subject to copyright
   60 :
  100 MODE7:DIM val(2,1)
  110 ON ERROR GOTO 340
  120 VDU23,1,0;0;0;0;
  130 val(0,0)=1:val(0,1)=2
  140 val(1,0)=4:val(1,1)=8
  150 val(2,0)=16:val(2,1)=64
  160 PROCforeground(0,0,3,150)
  170 PROCforeground(0,4,24,146)
  180 PROChline(4,75,1)
  190 PROCmsg2(6,1,130,"BEEBUG Teletext P
rocedures")
  200 PROChline(4,75,9)
  210 PROCframe(2,4,18,10)
  220 PROCframe(20,4,18,10)
  230 PROCframe(2,14,18,10)
  240 PROCframe(20,14,18,10)
  250 PROCbackground(22,5,12,131)
  260 PROCbackground(22,15,22,132)
  270 PROCmsg1(25,9,129,"RED TEXT     ")
  280 PROCmsg1(25,19,131,"YELLOW TEXT  ")
  290 PROCvline(21,13,70)
  300 PROChline(5,38,56)
  310 VDU28,3,13,18,5
  320 REPEAT:PROCpixel(RND(30),RND(22)):U
NTIL FALSE
  330 :
  340 MODE7:REPORT:PRINT" at line ";ERL
  350 END
  360 :
```

BRIGHTENING UP THE LANDSCAPE

For those of us who only have the humble model B, and use the 3D Landscape program from BEEBUG Vol.7 No.9, and are fed up with green/yellow fields, the extension below allows the full four-level green shading (as on the Master) with no expansion.

Type in the Master version of the program (or re-convert the BBC version). Make sure the move-down routine (lines 3000 to 3040) and line 105 are present. Edit line 180 to set N%=16. Then replace lines 2010 to 2070 with:

```
2010 DEF PROCshade(N%)
2020 IF N%=3 ?&359=&A
2030 IF N%=2 ?&359=&AF
2040 IF N%=1 ?&359=&F
2050 IF N%=0 ?&359=&A5
```

Delete line 2070, and if the 'No Room' error occurs on re-running the program, delete lines 20 to 100 as well. There may be slight 'dot crawl' on TV sets, but this is unavoidable. With luck the TV can be tuned to minimise or remove this problem.

Stephen Sexton

VERTICAL LINES IN PROGRAM LISTINGS

I have entered the DODECA game featured in BEEBUG Vol.7 No.10, but I am experiencing some difficulty in in entering line 3240, and in particular the vertical lines. What do I need to do to enter this line?

C.Green

The character to which Mr Green refers is the ***vertical bar*** *character with ASCII code 124. In our program listings this appears as a single vertical line, but on the screen (modes 0 to 6) it appears as a divided vertical line, and is also shown in this form on the computer's keyboard. The relevant key is to the left of the* ***cursor left*** *key and is a shift key operation. In mode 7, this character is shown on the screen as two parallel vertical lines.*

Mr Green is not alone in asking this question, which occurs more often than might be expected. Hopefully this will clear the whole matter up for all readers.

POSTBAG

UPDATING MAGSCAN

I recently purchased a copy of your Magscan for use with my Master 128. This was complete up to volume 6, and I have subsequently been trying to update it using View with the help of the volume 7 magazine discs (files MSCN701 etc.). So far I have been unable to do this successfully following the information in the instruction booklet which comes with Magscan. Can you help?

F.C.Wyllie

Using View, read in the file MSCN701 with the command:

```
READ MSCN701
```

In edit mode, delete the blank line at the top of the screen, and the ']' character at the end of the file. Now read in MSCN702 and repeat the process, doing the same for the remaining volume 7 updates. However, you must make sure that there is a ']' character at the very end of the complete file. Then save this file as ***Vol7****. You will also need to update the value assigned to N% in line 110 of the Magscan program to reflect the latest volume in use, '7' in this case.*

INDEXING WATFORD DOUBLE LENGTH CATALOGUES

Laurence Cox's useful utility for indexing discs (BEEBUG Vol.7 No.10) can be modified to read the 62 file double-length catalogue provided by Watford Electronic's Disc Filing system by altering and adding the following lines:

```
 1140 J%=(names?5) DIV 8
 1150 IF F%+J%+31>1000 THEN end=TRUE:ENDP
ROC
 1172 pblock?8=3:CALL start
 1174 J%=(names?5) DIV 8
 1176 pblock?8=2:CALL start
 1178 FOR I%=1 TO J%:F%=F%+1:FILE$(F%)=FN
readstring(8*I%,8)+"#"+STR$(T%):NEXT
```

Line 1140 corrects a bug in the original program which has already been published (Points Arising Vol.8 No.1 page 62).

L.Skilton

A number of readers have enquired about a modification to allow this program to work with the Watford DDFS, so we are pleased to be able to publish Mr.Skilton's update. B

free space list, thereby specifying how many free space entries there are, and byte 253 holds the boot option, as set by *OPT 4. Finally, bytes 251 and 252 contain the disc identifier. This is a sixteen-bit random number to ensure that ADFS can distinguish between discs enabling the system to give a 'disc changed' error. This is possible because it is very unlikely that any two discs will contain the same identifier.

The last byte on each of the free space map sectors holds a checksum on that sector. If this checksum ever becomes corrupted, ADFS knows that the information in the free space map cannot be relied on, and it therefore prevents any further write operations to the disc. The checksum for each sector is calculated in a rather unusual way, but the program below will calculate this for a 256-byte block of memory starting at the given address, and put the value in automatically. One very important consideration is that if you ever corrupt the free space map, most likely using a sector editor, then ADFS will not allow you to re-write a corrected version! However, our Disc Access ROM can write any sector to a disc, regardless of the free space map. Additionally, the ADFS Sector Editor featured in BEEBUG Vol.8 No.3 has a feature to automatically calculate the checksum.

WHAT USE IS IT?

You may be thinking that while all this information is vital to the filing systems themselves, it is of no practical use. However, this is not the case, because such information is essential when it comes to recovering accidentally deleted files, or worse still, files from a corrupted disc. In next month's BEEBUG we will take a look at this process, and give some hints to speed up the recovery process.

We present a further selection of hints and tips for the BBC micro, Master and Compact, rounded up by Mike Williams. Please continue to send in your contributions for this page. We pay £5 for each hint published and £15 for the star hint.

***** STAR HINT *****

PRINTING AFTER SCROLLING

Al Harwood

Scrolling the screen in any direction can be achieved by altering the contents of CRT registers 12 and 13. If you use this method to scroll, you cannot then immediately update the screen display, as the operating system has not been updated with the new screen position. To do this, you need to put the new address for screen top left in locations &350 (low byte) and &351 (high byte).

This is demonstrated in the following example, which uses the space bar to scroll the screen:

```
10 MODE 2:S%=HIMEM
20 PRINTTAB(0,5)" ... SCROLLING ..."
30 REPEAT:REPEAT:UNTIL GET=32
40 S%=S%+640:IF S%>&7FFF S%=HIMEM
50 VDU23,0,12,S% DIV 8 DIV 256
60 VDU23,0,13,S% DIV 8 MOD 256
70 ?&350=S% MOD 256:?&351=S% DIV 256
80 PRINTTAB(0,10)"I've just moved!"
90 UNTIL FALSE
```

S% is set to the address of that part of the screen display to appear in the top left corner of the screen, and this is incremented by 640 each time the screen scrolls (lines 40 to 60). Line 70 updates the system's knowledge of the screen's new position so that the PRINT statement in line 80 correctly positions the text string each time.

LINE SPACING ON EPSON PRINTERS

R.Hadekel

Printers are generally geared to the American 11 inches by 8.5 inches paper format, and with a default line spacing of 1/6 inch give 66 lines per page. To print at one and a half line spacing, issue the Epson control code sequence 27, 65, 18 (Escape A 18) giving a spacing of 18/72 inch = 1/4 inch or 44 lines per page (set your word processor to the same page length).

It is also possible to obtain approximately A4 size fan-fold paper, which allows 70 lines per page at 1/6 inch spacing, the paper sheet height being 11 2/3 inches. To obtain one and a half line spacing is more difficult. The Epson Escape sequence 27, 51,56 results in 56/216 inch line spacing giving exactly 45 lines per 11 2/3 inches page. This spacing of 56/216 inch = 7/27 inch which is quite close enough to 7/28 inch = 1/4 inch (needed for accurate one and a half line spacing).

ANSWERING THE USR CALL

J.Temple

The USR function in Basic can be used to call any machine code routine (particularly operating system routines) which return one or more values in registers. The USR function returns a single four byte number representing the contents of the accumulator, X register, Y register and status register. A simple way of separating these four values is to assign the returned value to memory location &70 (reserved for the user) by writing:

```
!&70=USR(address)
```

where *address* is the address of the routine being called. The accumulator contents will then be stored in &70, the X register in &71, the Y register in &72 and the contents of the status register in &73. These can be accessed with indirection operators, thus:

```
A%=?&70:X%=?&71:Y%=?&72:status=?&73
```

INSTANT ITALICS

Al Harwood

Use the following procedure in any program for instant italic text in modes 0 to 6:

```
1000 DEF PROCitalic(text$)
1010 LOCAL A,A%,A$,X%,Y%
1020 FOR A=1 TO LENtext$:A$=MID$(text$,A,1)
1030 ?&70=ASCA$:X%=&70:Y%=0
1040 A%=10:CALL &FFF1
1050 !&79=&6A0070B9:!&7D=&60007099
1060 FOR Y%=1 TO 8:IF Y%=4 ?&7C=&2A:Y%=6
1070 CALL &79:NEXT
1080 VDU23,128,?&71,?&72,?&73,?&74,?&75,
?&76,?&77,?&78,128
1090 NEXT:ENDPROC
```

The routine uses locations &70 to &80, and redefines the character with code 128. B

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 30p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS. The normal copy date for receipt of all ads will be the 15th of each month.

BBC B 1.2 OS, 40T DD, 32k Watford RAM, ATPL SWR board, AMX mouse, ROMs, Super Art, Pagemaker, View, Exmon II, Acorn GXR, Games £275 o.n.o. Tel. 01-635 8405 after 7pm.

Taxan Kaga KP810 NLQ printer with 6264 chip and Watford NLQ designer ROM/Disc, good condition, 8 assorted ribbons £150. WANTED: Master 512 board & kit Solidisk 1 meg if possible, serious request for educational project. Tel. (0767) 315443.

Mannesman Tally MT80+ (latest model including italics), BBC lead, manual and original packaging, £60 plus postage. Tel. (0932) 226076.

BBC Master 128, boxed in perfect condition. Purchased from BEEBUG and less than 1 year old. Includes manuals disc and leads £340. Tel. (0533) 712030 eves.

Games for model B, B+ and M128: IMOGEN (40T) £5, The Life of Repton (tape) £3.50, Play it Again Sam (tape) £5, all originals in original packaging and as new. Tel. (0742) 553418.

BBC B iss 7 with Watford DFS; 32k shadow RAM; Solderless ROM board; 2 DS 80/40 5.25" drives; Panasonic KX1082 printer; Zenith amber monitor; Quest mouse; Video Digitiser; plus View; ViewSheet; ViewStore; Inter Mega ROM; Spellmaster; View Printer ROM; Waping Editor; ADT; Dumpout 3 plus lots of disc based games and software £700 o.n.o. Will consider splitting. Tel. (0785) 713855 after 6pm.

BBC B with Torch Z80 disc pack and Torch Z80 Second Processor. CPN in ROM. Amber Monitor £250 o.n.o. Tel. (0656) 860781.

WANTED: 512 co processor. Tel. (0602) 201815 after 6pm.

Cumana double sided 80T disc drive with integral power supply £35. Tel. (0306) 889647 eves and weekends.

WANTED: 512k Co-Processor, also BEEBUG vols. 1,5,6 and 7. Tel. (0902) 783299 after 6pm.

M128, single 40/80T drive, DMP 3000 printer, Acorn Prestel adaptor, Music 500/5000 synthesiser, ROM cartridges, BEEBUG 'C' reference manuals, 30 discs many books and mags £600. Tel. (0202) 693301.

Business Ad

PEACOCK PRINTER

for multi-coloured printing from a BBC Microcomputer and an Epson-compatible monochrome printer.

£8.95 from DATASSISTANCE, 83 Main Street, Great Broughton, Cockermouth, Cumbria CA13 0YJ (0900 825503).

Please state 40 or 80 track disc.

BBC B 1.2 OS 40T disc drive, 32k Watford RAM, ATPL SWR board, AMX mouse, ROMS: Super Art, Pagemaker, View, Exmon II, Acorn GXR. Games £275 o.n.o. Tel. 01-635 8405 after 7pm.

BBC B issue 3 incl. ROM board with View, Viewsheet, Quest Paint (& mouse) PMS Genie, Dumpout 3, Stop Press & s/w RAM. Disc drive, Microvitec colour monitor, Nightingale modem. Microline printer free to buyer. Tel. (0952) 610489.

Brother EB44 portable typwriter/printer wp functions uses thermal paper or ribbon with paper roll. Mains transformer/Battery powered £125 o.n.o. Tel. (0722) 29341.

BBC B disc drive and monitor, choice of several machines DFS and disc drive types. Monitors available include Zenith Philips and Cub colour. Various ROMs available eg. Wordwise. Offers for systems should start around £250 Tel. 01-253 4399 extn 3275 or weekends (0487) 814227.

WANTED: EPROM blower or circuit diagram. Tel. (0442) 826079.

Watford ROM/RAM board 80K + battery £35, PMS 6502 second processor £35, Digimouse + driver £20. ROMs for model B or Master: GXR £10, PMS Producer £20, Wordwise + and Wordaid £20, Forth + manual £15, Advanced control panel £10. Tel. 01 388 0392 after 7pm.

Viglen 100K 40T SS D/D, good condition, complete with instruction manual £40. Tel (0727) 72865 after 6pm.

Epson RX80 F/T+ printer in good condition, complete with lead to connect to the BBC computer £100 o.n.o. Tel. (0332) 556381.

WANTED: 512K board + DOS 2.1+ DOS+ Problem Solver. Tel. 01-804 5984 after 4pm.

Vector 1&2 for BBC B tape to disc & disc to disc utilities £10, LCL Micro Maths for 0'Level revision. BBC & Electron £10, Mewsoft A4 Forms Designer 80T ADFS disc £6, Superior/Acornsoft EXILE new £8, Morley Master Smart cartridge new £20, Morley V2 EPROM programmer new £18, Morley EPROM utility disc new £3, 16K 21V EPROMs (each) 27128 £2.50, 8K 21V EPROMs (each) 2764 £1.30, Care Master cartridge Quad 4*16K new £8. Tel. (0784) 242817.

BBC B + Z80 + 512 + SRAM. 1770 DFS + ADFS + Twin switchable drives + speech. Philips CM 8833 colour/mono screen, Centronics GCP printer. Many manuals £600. Tel. (0570) 470763.

ISO Pascal £35 & Micro PROLOG £40, both original Acornsoft including all manuals, discs, box etc. Programming in Prolog made simple (book) £5, DS40T full-height uncased drive £50 o.n.o. WANTED: Watford 32k shadow RAM. Tel. (0707) 45953 eves.

Nidd Valley Digimouse + pointer ROM £20, INTER-BASE £30, INTER-SHEET £10, BROM+ utility ROM £11. Master 128 - Vines ROM overlay board £11, Dabs MOS+ ROM £5, Reference manuals 1 & 2 £11, ROM cartridge £4. Beebugsoft - Studio 8 £8, Masterfile II £8, Billboard £3, Teletext Pack £3 - all 5.25 discs. Games 5.25 disc - Elite/Exile/Life of Repton/Valley of Lost Souls £20 the lot. Tel. (0462) 682961.

WANTED: Interword ROM for BBC B with instructions. Also, JUKI 2200 daisy wheels eg Pica, Tile Narrator. Tel. (0903) 40531 after 5pm.

BBC Acorn DFS OS 1.2, 40/80 drive, Educational Software £250 Z80 processor, manuals, software, £150. Tel. (0883) 49657.

BBC B 128k, Watford twin D/D, Microvitec monitor Hybrid Music Synthesiser, Brother printer, ROM & disc software and books. Tel. 01-521 7456m,for full list or offers.

WANTED: ViewSheet User Guide. Tel. (0670) 860170.

BBC B OS 1.2 Watford DFS, Watford twin DS 40T D/D Watford Solderless ROM board incl. 2 off 6264 RAM Aries B20 Shadow RAM Philips 12" Green monitor £350 will split. Tel. 01-524 4239.

BBC B OS 1.2 with data recorder, tape games (Revs & Thrust) in box, joystick. Watford 3.5"/5.25" double sided 40/80T drive and MkII DDFS. Acorn ADFS, discs (Firetrack & Bonecruncher) books and magazines £420. Tel. (0302) 326084.

Master 128 inc MOS+, ADI, ACP, Hyperdriver, Genie, Quest paint +, mouse, 5.25" 80/40T drive, joystick + software, all manuals included, cartridges. £390 o.n.o. Tel. (0254) 61492.

BBC Master 512 (IBM compatible), Sanyo colour monitor, Twin 5.25" Technomatic drives in plinth, Twin 3.5" Cumana drives, AMX MkIII mouse, Joystick, PMS Genie cartridge, Epson LX86 dot matrix printer, Vine micros internal ROMboard, Viglen ROM cartridge system, many ROMs inc. Overview, Interword, BEEBUG Master ROM, many manuals and games software inc. Elite, Sentinel. Complete system worth over £1900. £1000 the lot. Tel. (0270) 666585.

Master 128 + 512K Co-processor, DOS+ Gem suite + mouse, DSDD dual 800k 5.25" disc drives, Microvitec 1451AP colour monitor, Brother HR15 printer, few games and books £900 the lot. Tel. (0689) 70132 after 5pm.

Master 512 with DOS+ V2.1, Acorn mouse, 5.25" & 3.5" disc drives, linked as a dual drive. Master reference manuals 1&2, Dabs Press-M512 User Guide + program disc, M512 Shareware collection, MOS+, Conversion kit, Sidewriter. Various other books, some with discs. BEEBUG C + standalone generator. Disc box, 5.25" discs £650. Panasonic KX-P1081 printer, incl. lead, £100. Tel. (0924) 826483 p&p extra.

AMX Super Art + Mk3 mouse £40, Master Turbo £70, Swivel Monitor stand 14" monitor £10, Viewplot £10 all as new. Tel. (04243) 4500.

Genie Junior plus utilities disc £16, ROMit for the BBC B £10. Tel. (0227) 369362.

WANTED: Floppywise Master and Brom Plus ROMs with manuals. Toolkit book by BBC Soft. Tel. (0294) 53648.

BBC B Viglen DFS and 40T SSDD drive, Philips BM7502 12" high res. green screen monitor, AMX mouse, Wordwise plus, B-Bore, filer, all manuals + advanced BBC user guide, discs cassettes and full set Beebug magazines £300 or will sell seperately. (0734) 302441 eves.

Star LC10 colour printer (4 months old) £190 with lead and two ribbons. Archimedes games: Minotaur £8, Startrader £10, Word Up Word Down £10, Clare's Arcade 3 £8, Terramex £12, Pacmania £12. Tel. 041-956 6106.

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES

		BEEBUG & RISC USER
£ 7.50	6 months (5 issues) UK only	£23.00
£14.50	1 year (10 issues) UK, BFPO, Ch.I	£33.00
£20.00	Rest of Europe & Eire	£40.00
£25.00	Middle East	£44.00
£27.00	Americas & Africa	£48.00
£29.00	Elsewhere	

BACK ISSUE PRICES (per issue)

Volume	Magazine	Tape	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.00	-	-
3	£0.70	£1.50	£3.50	-
4	£0.90	£2.00	£4.00	
5	£1.20	£2.50		
6	£1.30	£3.00		
7	£1.30			

All overseas items are sent airmail. We accept official for ck that handling orders under £10 which require an invoice. Note that there is no VAT in magazines.

See enclosed leaflet for Special Summer Offer on Back Issue magazines, tapes and discs!

Dest	Item
E	30p
	50p
	£1
El	£2

POST AND PACKING

Please add the cost of p&p as shown opposite.

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 60263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: David Spencer
Technical Assistant: Glynn Clements
Advertising: Sarah Shrive
Production Assistant: Sheila Stoneman
Membership secretary: Mandy Mileham
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

Printed by Newnorth-Burt Ltd (0234) 41111 ISSN - 0263 - 7561

Magazine Disc/Cassette

AUG/SEPT 1989 DISC/CASSETTE CONTENTS

A GENERAL PURPOSE LINE-INPUT FUNCTION - a highly versatile input routine for use in your own programs.

FONT DESIGNER (part 1) - a program allowing you to design and edit characters and complete fonts.

MATHEMATICAL TRANSFORMATIONS (part 2) - a complete program demonstrating mathematical transformations of shapes.

FIRST COURSE (Investigating Teletext Mode) - concluding the series on teletext with a program demonstrating some general purpose teletext procedures.

A SELF-HELP UTILITY - a utility to produce ROM images providing a keyword help facility.

ACES HIGH - a simple yet absorbing version of patience.

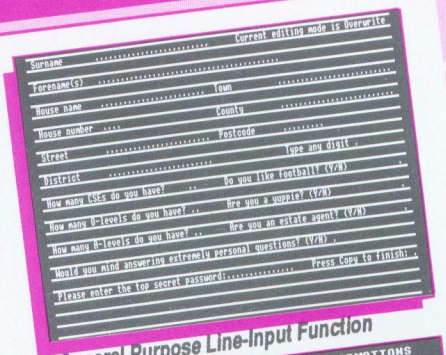
MULTI-COLOUR PRINTING - a utility to produce multi-colour print-outs using different coloured printer ribbons.

USING A VIDEO DIGITISER - a program which provides a front end to the Watford video digitiser and simplifies the set-up procedure.

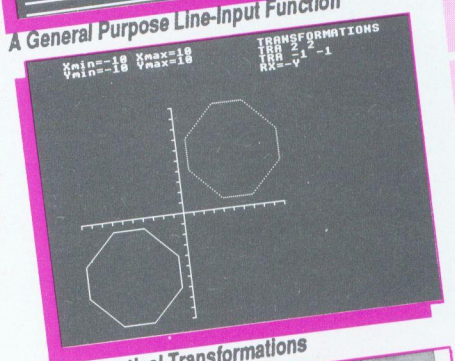
512 FORUM - a FIND utility for DOS Plus.

TELETEXT EDITOR (A Bonus Item) - a comprehensive utility for designing and editing teletext screens.

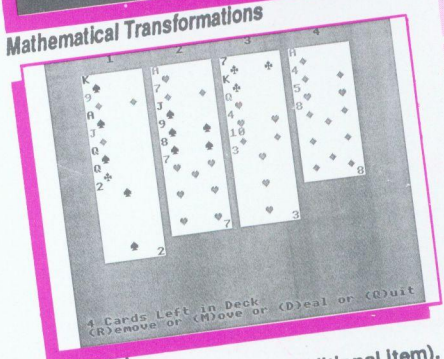
MAGSCAN - bibliography for this issue (Vol.8 No.4).



A General Purpose Line-Input Function



Mathematical Transformations



Aces High

All this for £3. 50 (cassette), £4.75 (5" & 3.5" disc) + 60p p&p (30p for each additional item).
Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

SUBSCRIPTION RATES	UK ONLY			OVERSEAS		
	5" Disc	3.5" Disc	Cassette	5" Disc	3.5" Disc	Cassette
6 months (5 issues)	£25.50	£25.50	£17.00	£30.00	£30.00	£20.00
12 months (10 issues)	£50.00	£50.00	£33.00	£56.00	£56.00	£39.00

Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:

BEEBUG, 117 Hatfield Road, St.Albans, Herts AL1 4JS.

FREE ON SITE MAINTENANCE

with all A3000's and Archimedes FROM BEEBUG

the Archimedes Specialists

Special Offers

If you subscribe to Beebug or Risc User, (or take out a new subscription) and purchase an Archimedes A3000 we will also supply: Artisan, Pacmania, Printer Lead, 10 3.5" discs and a Lockable Disc Box. In addition with a 410: First Word Plus, with a 420 First word Plus and PC Emulator and with a 440/1 First Word Plus, PC Emulator and the Software Developers Toolbox.

Speedy Service

We have a new showroom in St. Albans where you can see and try any of the systems. They are always on display and in stock. Our Mail Order Service is also very efficient, with over 80% of orders going out the same day. 48 hour fully insured delivery charges on complete system are just £7.00 and 24 hour delivery £11.50.

0% Finance

As a licensed credit broker, Beebug can offer you 0% finance over 9 months or 13.75% flat rate (typical APR 28%) over 12,24 or 36 months. Please phone to check the members special offers with 0% finance.

Technical Support

Unlike many dealers we don't lose interest as soon as you have made your purchase. You can always telephone our Technical Support Department or Showroom for some friendly and impartial advice.

Archimedes Magazine – RISC User

RISC User is the leading magazine dedicated solely to the Archimedes and A3000. For £14.50 a year (UK), you will receive this 56 page magazine packed full of useful information, news, reviews, programs, hints and tips etc; mailed directly to your home 10 times a year.

Ordering

All items are currently in stock. Prices are exclusive of VAT, please add 15%. We accept payment by cash, cheque, Visa, Access. Connect etc as well as official orders. You may order by phone, fax or letter. Alternatively please phone for more information.

COMPLETE CONFIDENCE

Quite simply this means that if your Archimedes goes wrong within the first year, an engineer will call at your premises within 24 hours to repair it. If it cannot be repaired, a replacement will normally be left.

NATIONWIDE COVERAGE

Beebug are using CRAY ELECTRONICS PLC to provide the maintenance cover which is available throughout the whole of mainland UK. Simply make one phone call and an engineer will call. There is no limit to the number of call-outs that you can make over the year.

ALL ARCHIMEDES & A3000

This service is included on all new A3000, A410, A420 and A440/1 systems sold by Beebug from September 10th 1989. It also covers the Acorn monitor if sold at the same time.

PRICES	BASE	COLOUR
A3000	649.00	869.00
A410/1	1199.00	1419.00
A420/1	1699.00	1919.00

PRICE GUARANTEE

We will match any current price or offer available from any other reputable Acorn dealer.

BEEBUG

117 Hatfield Road, St. Albans, Herts AL1 4JS.
Tel: 0727 40303 Fax: 0727 60263